A CISO's field guide to unified cloud access

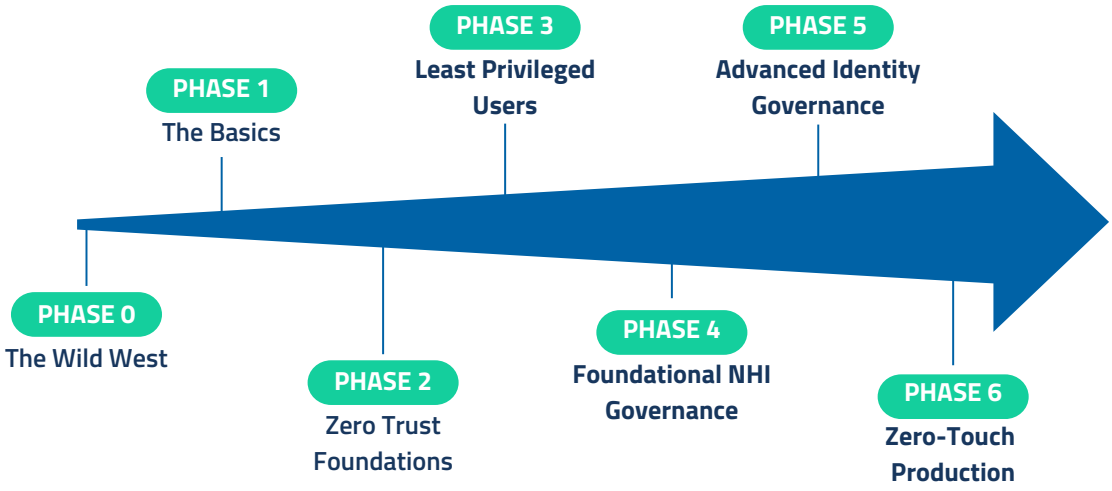
Introduction

Cloud identity is messy. Not because organizations are doing something wrong—but because the cloud is complex and inherently hard to secure. Today, the types of identities that can access the cloud are more diverse than ever: beyond human users to swaths of non-human identities like service accounts, IAM roles, service principals, and increasingly, AI agents. These identities can access the cloud in a vast number of ways—SSH, database sessions, Kubernetes shells, or thousands of cloud entitlements.

This explosion in access requirements has made orchestration, governance, visibility and risk posture management difficult.

But identity maturity isn't binary. You're not simply "secure" or "insecure." Most organizations aren't in a single phase—they're in multiple, depending on the system or environment. You might have strong governance over production cloud infrastructure, but weak controls around corporate apps that gate access to that cloud. That's normal.

This guide is here to help you identify where you are—across domains—so you can decide where to go next. It's not a checklist. It's a map. And while your destination might not be zero-touch production, understanding where you stand—and what tradeoffs got you there—is the first step toward something more sustainable.



Contents

PHASE 0: The Wild West

"Security is an afterthought. If at all."

This is the phase no one wants to admit they're in—but most teams have passed through, and some never really leave.

In Phase 0, organizations haven't secured any meaningful layer of identity, authentication, or access in the cloud. Often, it's because they're just beginning their cloud journey—or because security still isn't seen as critical to operations. We all hope such companies are rare—or at least urgently investing to move further right on the maturity curve.

Security teams here are either nonexistent or drowning. Access is handed out like Halloween candy. Developers share credentials to unblock a deployment or onboard a new teammate. There's no audit trail—only tribal knowledge.

Identity and access governance doesn't just lack enforcement—it lacks structure. IT might be creating some cloud accounts, while engineering or DevOps hand out others. There's no single view of who has access, or how it's granted. Shadow IAM becomes the norm, not the exception.

Engineers log into cloud environments as local users. SSO and MFA are wish-list items. Secrets float through Slack threads and network shares. SSH keys are emailed. Credentials are shared "just for now" and never revoked. Everyone's moving fast and breaking things—but access isn't one of them. It's just... there.

This isn't just technical debt. It's cultural debt. IAM is someone else's job—or worse, a blocker. There are no approvals, no offboarding, no expiration. Permissions accumulate. Like rust.

You know you're in Phase 0 if:

- SSO and MFA are seen as luxuries
- No one can tell you who has what level of access to where.
- The default assumption is that everyone has admin privileges
- Secrets are embedded in Terraform, CI/CD jobs, and README files.

What keeps you here:

- Speed culture: IAM is always tomorrow's problem.
- Lack of ownership: No one wants to be "the IAM person."
- Fear of breaking things: "Don't touch the keys—it just works."
- Security is an afterthought: Security might not be critical to business

"We lost a senior engineer last week. He had access to everything—Snowflake, prod clusters, GitHub org. I think we revoked most of it… but I'm not totally sure."

This phase isn't a failure. It's a reflection of organizational reality. Access is granted because people need to move. Governance is absent because no one has time. It's a tradeoff. A risk. Sometimes, a calculated one.

But it doesn't scale.

What pulls you out of Phase 0:

- A new hire with scars. Someone joins who's been burned by poor access controls and knows how to build better ones.
- A moment of friction. A prod incident tied to overprivileged access, stale keys or exposed secrets.
- The first real audit. It doesn't even have to be SOC 2. Just a security-minded customer asking questions you can't easily answer.
- Organic scale. As the team grows, shared secrets and ad hoc access stop being "fast"—and start being a liability.

How to move forward:

Establish basic hygiene. For the most part, you don't need a lot of advanced tooling yet. Just awareness. A spreadsheet might be enough process to know who can do what, make sure credentials can be changed on departure, and to get you through that first audit.

PHASE 1: The Basics

"We turned on SSO and MFA. We're good, right?"

This is the phase where an organization starts to notice and consider identity and access. There's an identity provider in place, and SSO + strong authentication like FIDO2/TouchId to the cloud console. Cloud access might be gated behind a VPN, or a Zero Trust gateway such as Zscaler or Cisco Umbrella. It looks secure from a distance—and at least, it's no longer the Wild West.

But hygiene doesn't mean safety. It means you're less obviously vulnerable.

While basic authentication is in place, access is still largely unmanaged. VPNs might sound safe, but they create unnecessary friction for developers. SSH keys are still floating around, but now they're "uploaded to the vault" instead of stored in a text file and rarely rotated. Roles are coarse-grained: admin, dev, read-only—and applied broadly.

It's like putting a lock on the front door, but leaving every window open. SSO helps with who can log in. It doesn't help with what they can do once they're inside.

In Phase 1, teams often feel like they've already "done IAM." The checkboxes are marked. The audit passed. But access reviews are spreadsheet-based, access revocation is manual, and off-boarding still requires tribal knowledge. Governance is reactive, not embedded.

This phase is safer than Phase 0, but not resilient. And it won't hold under scale.

You know you're in Phase 1 if:

- You're using a VPN to connect to the cloud, but do not have a good process for rotating SSH keys. You might be using a vault, but not enforcing secrets rotation.
- You don't have a complete inventory of who has access to which systems. Your governance process is a shared Google Sheet and a calendar reminder.
- Access is provisioned based on static and overprovisioned roles.

What keeps you here:

- The illusion of safety. You've checked some important boxes—why push further?
- Tools without process. You have the right infrastructure, but no one owns policy design.
- Fear of complexity. Least privilege feels abstract. Who has time to re-architect access?

What pulls you out of Phase 1:

- ssh keys are shared amongst engineers, not rotated periodically, and one day, one such key might lead to a production incident
- A customer, auditor or partner asks for proof of fine-grained access controls—and you realize you can't answer.
- A shift in culture—someone on the team starts asking: "Why does this role have production access by default?"

How to move forward:

- Start breaking access into smaller, more specific scopes. Identify your most privileged roles and reduce their blast radius. Begin implementing keyless workflows for infrastructure access. Track who has access to what—not just who logs in.
- This is where IAM becomes not just about access, but about control.

PHASE 2: Zero Trust Foundations

"We replaced the VPN. Now what?"

At this stage, things start feeling more modern—organizations typically begin reconsidering or supplementing VPN access. VPNs have historically been the default remote-access solution, but many teams now experiment with alternatives such as bastion hosts, jump boxes, and keyless access via short-lived certificates or federated identity.

However, the shift away from VPNs doesn't automatically mean a stronger security posture. True Zero Trust architectures evaluate every access request individually, rather than granting implicit trust based on network boundaries. In practice, though, Zero Trust adoption is often incomplete. Bastion hosts frequently retain standing privileges, access policies remain overly broad, and persistent SSH access to critical systems persists. Early employees and founders often retain legacy access rights. And secrets—those haven't disappeared either. They've just been vaulted and forgotten.

In other words, changing your access method isn't enough. Zero Trust is fundamentally about minimizing implicit trust, continuously verifying permissions, and rigorously enforcing least privilege, regardless of the technology choice.

Phase 2 feels like progress because it is. But it's also where many teams stall. The architecture has evolved, but governance hasn't caught up. The move from network security to identity-first access control is incomplete—and the friction between security and developer velocity starts to show.

This is the phase where Zero Trust marketing collides with cloud reality. And it's where you start realizing that least privilege is not a philosophy. It's an implementation detail. A hard one.

You know you're in Phase 2 if:

- Bastion hosts have replaced your VPN, but your developers still have standing/persistent access to the cloud.
- You might have experimented with Identity-aware proxies instead of traditional bastion hosts—to centralize authentication and reduce direct SSH access. However, the underlying challenge of managing standing privileges remains.
- Access continues to be provisioned based on broad, static and overprovisioned roles.

What keeps you here:

- Bastions are "good enough." They work, and no one wants to rebuild that flow again.
- Dev teams push back on anything that might block access to infrastructure, or otherwise introduce any friction in their jobs.
- No one owns the question of "should this person/service have this access right now?"

What pulls you out of Phase 2:

- A high-profile incident where standing access is implicated—whether internal or seen in the news.
- An internal champion who frames fine-grained access as a developer experience upgrade, not a security constraint.
- A compliance requirement for per-resource, per-user or per-session access review.

How to move forward:

Start exploring JIT (just-in-time), short-lived and least-privileged access. Introduce ephemeral credentials for infrastructure access—issued via Slack or CLI with clear time bounds. Begin moving away from static IAM roles toward scoped, context-aware permissions.

Least privilege doesn't begin with a policy. It begins with visibility.

PHASE 3: Least-Privileged Users

"We finally know who has access to what. Sort of."

This is the phase where visibility clicks in—for the first time, teams start to get a real picture of human identity sprawl. CIEM or CSPM tools are introduced. Dashboards light up with insights: who's overprivileged, who holds admin access, which identities haven't touched a key system in months. The fog lifts—revealing a mess.

With this visibility comes a sense of urgency. Organizations start implementing access reviews. Some begin experimenting with **just-in-time access for human users**, often scoped to broad roles like read, write, or admin. A few go further, attempting **fine-grained entitlements**: this engineer can read only this S3 bucket, or restart just this Kubernetes cluster.

It's a real inflection point. Permissions are no longer monolithic. Policies are no longer implicit. But this progress brings friction. Access decisions now involve context, edge cases, and escalation paths. And the tooling? Often still manual and inconsistent.

The key here: visibility into human access is finally possible. Governance becomes less about trusting group membership, and more about validating actual need. But non-human identities remain largely invisible. Secrets, service accounts, and automation tokens are still out of frame—that's what Phase 4 is for.

You know you're in Phase 3 if:

- You've adopted a CIEM or CSPM to surface identity risk.
- You run scheduled access reviews, even if they're manual.
- You implement JIT access for cloud identities.
- You've started breaking roles into narrower scopes, tied to environments or resource types, e.g. a role that gives engineers read access to sensitive S3 buckets in production, as compared to a broader admin role.

What keeps you here:

- Governing user access is considered good enough, and is sufficient to satisfy most auditors.
- JIT workflows are generally a good developer experience, and you are hesitant to modify something that is working well.
- Visibility provided by CIEM tools can get very granular and generate a lot of alerts. You may not know how to proceed and set up the right least-privileged roles for your engineers

What pulls you out of Phase 3:

- Your developers are happy, but your security team is drowning in managing entitlements, roles, and edge cases—keeping up with your business growth and changes.
- Auditors start asking questions about service accounts and IAM roles—and you realize that you don't know the purpose behind 90% of those NHIs.
- Secret sprawl: secrets end up in logs—or in GitHub—and no one notices until it's too late. Or a dev team ships a new service with hardcoded credentials, with no governance process in place.
- Someone asks, "What's our inventory of non-human identities?" and the answer is a shared spreadsheet.
- Your scale forces a shift toward an **"assume breach"** mindset. You recognize that if an attacker gets inside your perimeter—through phishing, token theft, or a compromised workload—they won't need to exploit vulnerabilities. They'll simply **abuse the standing permissions** and **over-provisioned identities** already present in your environment.

Without stronger access controls, every identity becomes a potential pivot point.

How to move forward:

Shift your focus from human access to machine access. Begin treating service accounts, tokens, and secrets with the same urgency as users and user credentials. Establish a source of truth for all non-human identities. Build a process for key rotation that doesn't rely on engineers remembering to do it.

Next up: Phase 4—where the machines take over. Let's go.

PHASE 4: Foundational NHI Governance

"We've secured humans. The bots are winning."

By this phase, human access is under reasonable control. But non-human identities—service accounts, IAM roles, service principals, AI agents, etc.—now outnumber your human users 100 to 1. And they have more privileges. This is where organizations realize: the problem isn't just who has access. It's what has access—and how.

The move towards NHI governance starts with visibility. The first step is to generate a full inventory of all NHIs in your environment, the associated credentials and secrets, and the human owner for each NHI. Once organizations have this visibility, the next step is to implement least privilege for NHIs and reduce reliance on static/long-lived credentials.

Implementing least privilege for NHIs introduces significant operational risk. Every change brings the potential of breaking critical production services, creating downtime, or causing outages. This risk is magnified because achieving least privilege typically requires modifications to application code—a demanding task for engineering teams that consumes substantial resources. The tradeoff isn't hypothetical anymore. It's real.

Reducing reliance on static credentials is even more difficult. In theory, the best practice is ensuring every authentication uses short-lived dynamic credentials. But in practice, long-lived keys are everywhere. Secrets are passed between systems, stored in environment variables, and embedded in configuration files. Many haven't rotated in years. Few are tied to a single use. Most are invisible to access reviews. Converting them all to short-lived authentication methods is currently a pipe dream. Most organizations stop at automating secret rotation.

You know you're in Phase 4 if:

- You're building or buying infrastructure for secret rotation.
- You catch yourselves minting new services, with the same old shared role
- Your platform team has begun rotating secrets and trimming permissions for NHIs - but breaking things when they do.

What keeps you here:

- Secrets are everywhere. You have rotation tooling—but no consistent rollout or testing strategy, and the cost to fix is massive.
- Engineering resists changes that slow delivery or add risk.
- IAM roles for NHIs are still broad ("admin" or "read/write" at the project level), and it will likely take a lot of resources to implement least privilege for NHIs

What pulls you out of Phase 4:

- A token leak leads to real damage, or a near miss.
- You adopt a platform team with a mandate, and more importantly, sufficient bandwidth, to secure NHI workflows.
- Your incident response playbook starts with "rotate everything"—and that's no longer viable, due to the sheer number of NHIs.

How to move forward:

- Automated secrets rotation, but wherever possible, replace static credentials with short-lived federated credentials for workloads.
- Begin enforcing lifecycle and ownership on non-human identities.
- By all means, rotate secrets and trim workload permission, but make the changes testable, observable, and rollback-friendly.

Machines aren't the exception. They're the norm. And they're your biggest identity surface.

PHASE 5: Advanced Identity Governance

"We automated everything. Now we're afraid to touch it."

This is where the environment starts to feel truly modern—maybe even elegant. Automatic secrets rotation and least-privileged NHIs become table-stakes. And so organizations now begin to move past long-lived credentials altogether. Developers start moving services to a centralized workload auth, that relies on short-lived certificates or tokens.

And yet... the fear sets in.

Because now, every change is high stakes.

The platform is mature, but fragile. Organizations are likely using JIT access to fine grained permissions, scoped at the level of individual resources (such as "give me read access to a specific S3 bucket"). Permissions are finely scoped, but brittle. A single misconfigured policy can take down production. A secret rotation job that fails silently can lock out an entire CI/CD pipeline. And even with automation, you're constantly playing catch-up with drift—resources being created outside of code, roles accumulating unintended permissions, temporary hacks that become permanent.

The gap between your desired state and your actual state starts to widen again. And the blast radius has never been higher. Organizations hence start implementing policies for drift detection and remediation.

You know you're in Phase 5 if:

- Every workload identity uses short-lived, scoped credentials by default.
- Access decisions are automated based on context—user, role, device, workload.
- Drift detection is active and alerts are routed—but triage still feels manual.
- IAM and secrets are managed as code—but enforcing policy drift is still fragile.

What keeps you here:

- The tooling is ahead of the org. You've built the plane—but not everyone knows how to fly it.
- Every change now comes with risk. Automation removes the human—but not the consequences.
- You have the right policies, but not the observability to prove they're working.

What pulls you out of Phase 5:

- An outage tied to a failed credential rotation or permission regression.
- Engineers pushing for safer ways to test permission changes before rollout.
- Organizational pressure to use automation to reduce overhead without increasing fragility.

How to move forward:

Automation. Validation. Observability.

Treat IAM as part of your release pipeline. Automate permission testing. Create observability for drift—not just alerts, but context. Use canary-style rollout for secrets and access changes. Minimize standing access and minimize fear of changing access.

You've built a zero-trust system. Now you have to trust it.

And now, Phase 6!

PHASE 6: Zero-Touch Production

"There's no standing access. And no one misses it."

This is the dream. No persistent credentials. No long-lived tokens. No humans with production access by default. Every access – for humans and machines alike – is short-lived, keyless, just-in-time and least-privileged.

Secrets rotate hourly, or are eliminated entirely. Access is scoped by context, environment, resource and intent. Policies are enforced at runtime. And developers can still ship. Fast.

This phase is about precision at scale. It's where automation isn't just helpful—it's essential. Infrastructure-as-code and identity-as-code pipelines are humming. Nothing is manual. Everything is declarative.

This phase is real. But it's rare.

You'll see it in hyperscalers. Maybe in elite security engineering teams at FAANG companies. Sometimes in tightly scoped internal platforms with deep DevSecOps investment. But for most orgs, Phase 6 isn't a destination. It's a North Star.

That's okay.

Zero-touch production is a great aspiration—but often an unrealistic goal. What matters more is why you'd pursue it. Not because it's shiny. But because it enforces rigor, removes guesswork, and removes people from the blast radius entirely.

You know you're in Phase 6 if:

- You can't log into production—not because you're blocked, but because you don't need to.
- Access is fully ephemeral for both humans and machines.
- Credentials are issued based on policy, not requests.
- Drift is automatically detected and remediated—no manual intervention required.

What keeps you here:

- A security team with engineering capacity—and executive support
- A developer experience that feels native, not bolted-on.
- Heavy investments in platform, security and automation.

What pulls you here:

- You don't get pulled here. You build here. Carefully. Over time. With culture, code and coordination.

Zero-touch isn't about removing people from the loop. It's about building a loop that doesn't need them.

Cloud Access Maturity Assessment

Knowing your maturity phase isn't about where you aspire to be—it's about where you actually are today. This checklist is designed to help security, platform, and DevOps teams quickly identify where they fall along the Cloud IAM Maturity Curve.

It's not a scorecard. It's a mirror.

Use it to spark conversation, highlight gaps, or assess how fragmented (or redundant) your current tools might be.

If nothing else, it'll help you name what's working—and what's in need of a rethink.

Let's get started...

Phase	Signs You're Here	Tools You're Likely Using	Gaps to Watch
PHASE 0: Wild West	• No SSO/MFA • SSH keys passed around manually • No formal offboarding • Secrets in code, Slack, CI/CD	None or ad hoc scripts, manual Terraform Maybe 1Password, Slack pins	No visibility No rotation No accountability
PHASE 1: The Basics	• SSO and MFA enabled VPN still in use • Broad IAM roles (admin/dev) • Vault exists but isn't enforced	Okta, Google Workspace, GitHub SSO VPN (OpenVPN, WireGuard), Zscaler, HashiCorp Vault, or AWS Secrets Manager	Shared ssh keys Static keys Coarse-grained access Offboarding still ad hoc
PHASE 2: Zero Trust Foundations	• Bastion hosts replace VPN • Standing access persists • Short-lived creds for humans • Role definitions still manual	Teleport, StrongDM, AWS Session Manager	Standing access Secrets unmanaged
PHASE 3: Least-Privileged Users	• CIEM/CSPM alerts on overprivilege • Manual access reviews • Fine grained roles and policies • Least privileged, JIT, short-lived access to prod	Wiz, Prisma Cloud, Okta Workflows Manual spreadsheets	Alert fatigue No enforcement NHI access still invisible
PHASE 4: Foundational NHI Goverance	• Inventory of NHIs • Secrets rotation in place • Least privileges for NHIs	Vault, AWS/GCP Secrets Manager, in-house tooling for rotating secrets	Code change fatigue Sprawl of static creds No central lifecycle for NHIs
PHASE 5: Advanced Identity Governance	• All creds ephemeral • Drift detection built-in • Permissions managed as code • Observability across human/NH	IAM-as-code systems (Terraform modules) Custom scripts Internal dashboards Centralized/federated workload authentication	High fragility Rollout risks Orgs can't keep up with complexity
PHASE 6: Zero-Touch Production	• No standing access • Access fully scoped, ephemeral • Secrets auto-rotated + remediated	Google-style internal platforms Custom IAM engines Privileged access policies fully automated	Requires differentiated prioritization and investment, typically seen in hyperscalers

Phase	Signs You're Here	Tools You're Likely Using	Gaps to Watch
PHASE 0: Wild West	• No SSO/MFA • SSH keys passed around manually • No formal offboarding • Secrets in code, • Slack, CI/CD	None or ad hoc scripts, manual Terraform Maybe 1Password, Slack pins	No visibility No rotation No accountability
PHASE 1: The Basics	• SSO and MFA enabled VPN still in use • Broad IAM roles (admin/dev) • Vault exists but isn't enforced	Okta, Google Workspace, GitHub SSO VPN (OpenVPN, WireGuard), Zscaler, HashiCorp Vault, or AWS Secrets Manager	Shared ssh keys Static keys Coarse-grained access Offboarding still ad hoc
PHASE 2: Zero Trust Foundations	• Bastion hosts.replace VPN • Standing access persists • Short-lived creds for humans • Role definitions still manual	Teleport, StrongDM, AWS Session Manager	Standing access Secrets unmanaged
PHASE 3: Least-Privileged Users	• CIEM/CSPM alerts on overprivilege • Manual access reviews • Fine grained roles and policies • Least privileged, JIT, short-lived access to prod	Wiz, Prisma Cloud, Okta Workflows Manual spreadsheets	Alert fatigue No enforcement NHI access still invisible
PHASE 4: Foundational NHI Goverance	• Inventory of NHIs • Secrets rotation in place • Least privileges for NHIs	Vault, AWS/GCP Secrets Manager, in-house tooling for rotating secrets	Code change fatigue Sprawl of static creds No central lifecycle for NHIs
PHASE 5: Advanced Identity Governance	• All creds ephemeral • Drift detection built-in • Permissions managed as code • Observability across human/NH	IAM-as-code systems (Terraform modules) Custom scripts Internal dashboards Centralized/federated workload authentication	High fragility Rollout risks Orgs can't keep up with complexity
PHASE 6: Zero-Touch Production	• No standing access • Access fully scoped, ephemeral • Secrets auto-rotated + remediated	Google-style internal platforms Custom IAM engines Privileged access policies fully automated	Requires differentiated prioritization and investment, typically seen in hyperscalers

Co-authors

Adeel Khurshid
Senior Director of Security Engineering, **Splunk**

Adeel is a seasoned infrastructure and security leader with deep expertise in site reliability engineering, production infrastructure, and information security. Over the past two decades, he has played pivotal roles at some of the most respected technology organizations in the world—including Meta, GitHub, and currently Splunk, where he serves as **Senior Director of Security Engineering.**

At Meta, Adeel led production engineering for critical Security and Integrity infrastructure, including SIEM/SOAR platforms and content moderation systems at global scale. Prior to that, he served as **Director of Security Infrastructure and Operations** at GitHub, where he was responsible for IAM, secrets management, vulnerability management, and cloud security operations during a period of rapid organizational and platform growth.

Adeel has a proven track record of building high-performing, globally distributed engineering and security teams. At Radius Intelligence, he served as **VP of Technology and CISO**, where he built an end-to-end information security program from the ground up, aligning the company's operations with compliance and enterprise-readiness.

He brings both technical and business acumen to his leadership approach, underpinned by an MBA in Managing Innovation and Technology from **Santa Clara University** and a B.S. in Computer Engineering from **Old Dominion University**. He also holds the **Certified Information Security Manager (CISM)** from ISACA.

Shashwat Sehgal
FOUNDER AND CEO, **P0 Security**

P0 Security Is a venture-backed cybersecurity company pioneering a unified platform for cloud identity governance and access management. Under Shashwat's leadership, P0 Security has quickly emerged as one of the most innovative startups in the space, named to the Fortune Cyber 60 list and selected as a finalist for the RSA Innovation Sandbox.

P0 was born from a clear insight: modern cloud environments have outgrown traditional PAM, IAM and IGA tooling. Drawing from deep firsthand experience leading product at companies like Splunk and Cisco Meraki, Shashwat recognized that developers were building faster than security could govern. The result was a visibility and access control problem that spanned human and machine identities across AWS, GCP, and Azure. P0 solves this by providing real-time access orchestration, risk-aware governance, and automated remediation—all with a developer-first experience.

Prior to founding P0, Shashwat led the **Digital Experience Monitoring (DEM)** product portfolio at **Splunk**, launching and scaling products like browser and mobile RUM and leading M&A integrations to expand the observability suite. Before that, he served as **Group Product Manager at Cisco Meraki**, where he built and scaled Meraki Insight from concept to $20M+ ARR and managed the SD-WAN portfolio through a period of hypergrowth.

Earlier in his career, Shashwat was a strategy consultant at Booz & Company and a quant/engineer at Merrill Lynch in New York and London—roles that shaped his analytical edge and systems-thinking approach.

Shashwat holds a **Bachelor of Technology in Computer Science** from I**IT Delhi**, where he graduated with the Director's Gold Medal, and an **MBA from Columbia Business School**.

P0 SECURITY

Contact Us

Email: info@p0.dev
Web: www.p0.dev

P0 Security is the unified access privilege platform built for modern cloud infrastructure. Where legacy IAM, PAM, and IGA tools fall short — particularly around non-human identities, ephemeral infrastructure and developer velocity — P0 delivers orchestration and governance, visibility and risk posture across all cloud environments.

With an agentless, API-native architecture, P0 helps teams enforce least privilege by default through short-lived, just-in-time access for both human and machine identities. Security and platform teams rely on P0 to reduce blast radius, streamline audits and eliminate manual provisioning without slowing down development.

P0 is trusted by leading organizations in fintech, healthcare, AI, and cloud-native tech, with full enterprise deployments completed in under 60 days. Learn more at www.p0security.dev.