

Buyer's Guide for Secure Infrastructure Access

Table of contents

Executive Summary	3
Who This Guide Is For	4
Part One – The Problem and the Category	5
What Secure Infrastructure Access Means	5
How this relates to what you already run	6
Why the category name keeps moving	7
What Breaks Without It	7
AI Agents and Infrastructure Access	9
What Is Driving Urgency Now	10
Part Two – Defining What You Need	11
Step 1. Map Your Infrastructure	11
Step 2. Map Everything That Needs Access	12
Step 3. Define Your Requirements	13
Step 4. Build the Business Case	16
Part Three – Evaluating, Selecting, and Deploying	17
Step 5. Understand the Approaches	17
Step 6. Run the Proof of Concept	19
Step 7. Deploy	20
Step 8. Reach Steady State	21
Part Four – How Delinea Approaches This	23
Appendices	24
A. Vendor Question Bank	24
B. Requirements Checklist and Scoring Template	26
C. How Evaluations Go Wrong	27
D. Glossary	28

Executive Summary

Most companies running production infrastructure have already partially solved this problem three or four times. A credential vault holds the passwords that matter, a VPN gets people onto the network, and bastion hosts front the Linux fleet. Somewhere there is a Terraform module that provisions database users, and every one of those decisions made sense at the time. Together they add up to a set of overlapping controls with seams between them, and attackers, auditors, and brand-new engineers all find the same seams.

This guide covers how to evaluate and buy a solution that closes those seams. Secure infrastructure access means one control point in front of the systems that run your business. Behind that control point sits a single policy governing every identity that requests access. Employees, contractors, service accounts, and AI agents all answer to the same rules.

Five decisions determine whether the project works:

- What counts as infrastructure in your environment, and what lands in phase one
- Whether authorization happens only when a session opens, or continuously while it runs
- Who authors access policy, and who reviews it once the project team disbands
- What an auditor can retrieve without your engineers assembling it
- What this sits alongside in your stack, and what it lets you turn off

The eight steps that follow run from inventory through steady state. Appendix B holds a scored requirements checklist, and Appendix A holds the questions to ask on a vendor call. If you read nothing else, read Step 3 and take the checklist into your first demo.

One warning before you start. Most failed evaluations in this category fail for reasons that have little to do with technology. The POC covered one protocol and never touched the rest of the estate. Nobody invited the DBA team, and nobody built the cost model, so procurement sat on the deal for two quarters. Steps 4 and 6 exist because of that.

FIVE DECISIONS DETERMINE WHETHER THE PROJECT WORKS

- 01 Scope**
What counts as infrastructure, and what lands in phase one
- 02 Authorization**
At session open only, or continuously while it runs
- 03 Policy ownership**
Who authors it, and who reviews it after the project team disbands
- 04 Evidence**
What an auditor retrieves without engineers assembling it
- 05 Coexistence**
What it sits alongside, and what it lets you turn off

Who This Guide Is For

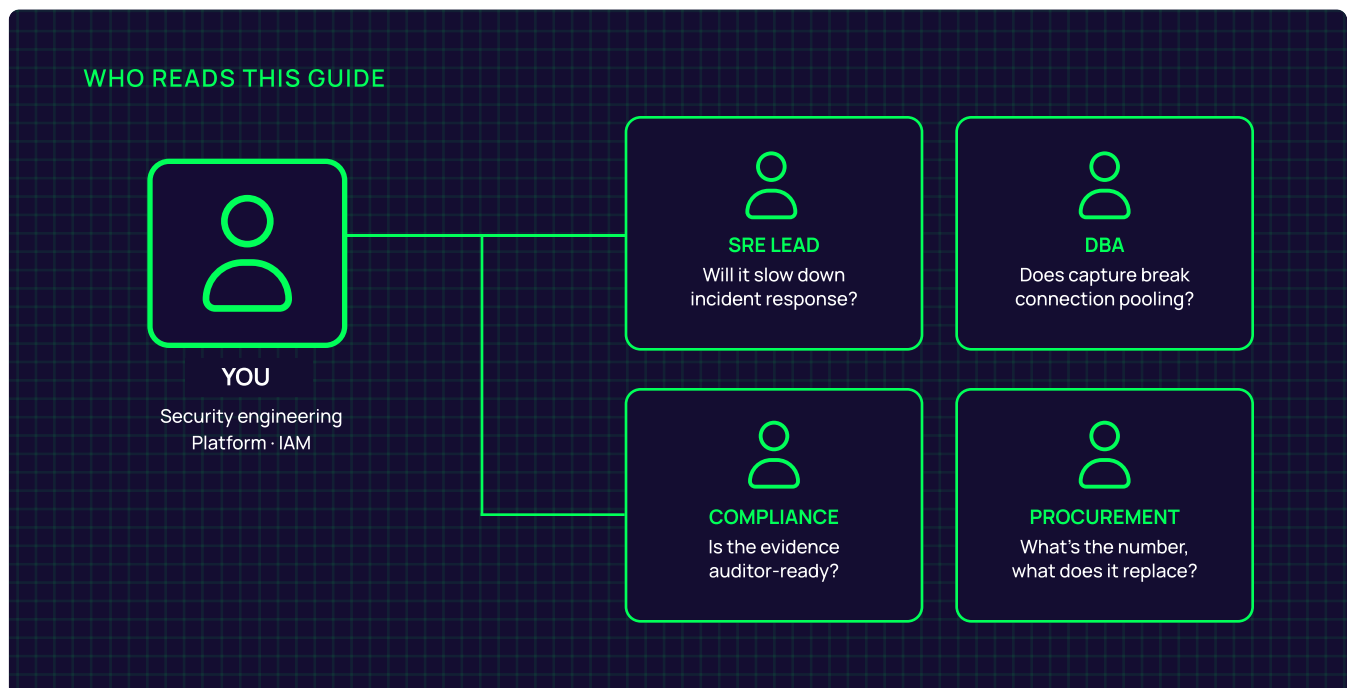
You own access to production infrastructure, and you have concluded that the current arrangement will not survive the next audit, the next headcount jump, or the next platform migration. Usually that comes with a title like director of security engineering, head of platform, or IAM architect.

Four other people read over your shoulder, and this guide gives each of them something. The SRE lead wants to know whether this slows down incident response, and the DBA wants to know whether query-level logging breaks connection pooling. The compliance owner wants evidence they can hand to an auditor. Procurement wants the number and what it replaces.

The guide assumes you already run an identity provider, you have some form of credential vault or PAM in place, and more than a handful of engineers touch production. Below that scale, bastion hosts and discipline still carry you for a while. The practical sequence is to close the inventory and offboarding gaps first, then revisit this decision when engineering headcount, audit scope, or agent adoption moves.

It does not cover workforce SSO for SaaS applications, endpoint privilege management, cloud infrastructure entitlement management, or application-to-application secrets. Those are adjacent, they matter, and they are separate buying decisions.

Three sections include a “From the Field” note. Those are first-hand accounts from a Delinea Field CISO who previously ran privileged access globally for a multinational petroleum company. Each one covers the part that never appears in a project plan.



PART ONE

The Problem and the Category

What Secure Infrastructure Access Means

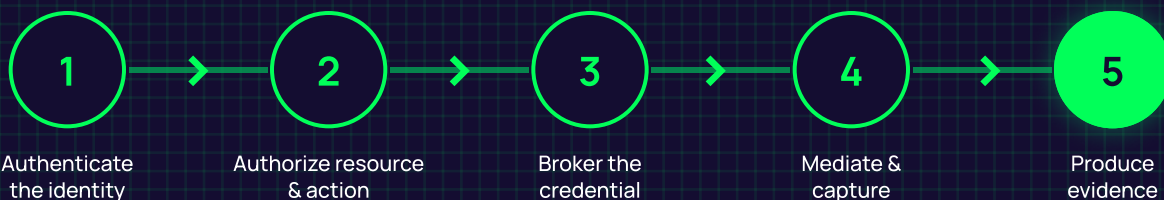
Infrastructure here refers to the systems your engineers directly manage to keep the business running. That covers Linux and Windows servers, Kubernetes clusters, relational databases such as PostgreSQL, MySQL, SQL Server, and Oracle, data warehouses, cloud consoles and APIs, message brokers, MCP servers, internal admin applications, network gear, and whatever legacy box still runs the nightly billing job.

Those systems share three traits that set them apart from the rest of the application estate. Access occurs at the protocol level via SSH, RDP, kubectl, psql, and raw HTTPS, rather than through a browser SSO flow. The credentials are technical, often local to the system, and frequently shared across a team. The blast radius of one bad session is also far larger, since a person with the wrong database privileges can cause more damage in 10 minutes than a compromised SaaS account can in a month.

Secure infrastructure access covers five jobs:

- Authenticating the identity making the request
- Authorizing the specific resource and the specific action
- Handling the credential so the requester never holds it
- Mediating and capturing the session
- Producing evidence somebody outside your team can verify

SECURE INFRASTRUCTURE ACCESS DOES FIVE JOBS



How this relates to what you already run

PAM. Privileged access management owns the credential lifecycle. It vaults, rotates, and brokers the accounts carrying the most risk, and applies session management to administrative access. PAM answers where a credential lives and who may use it.

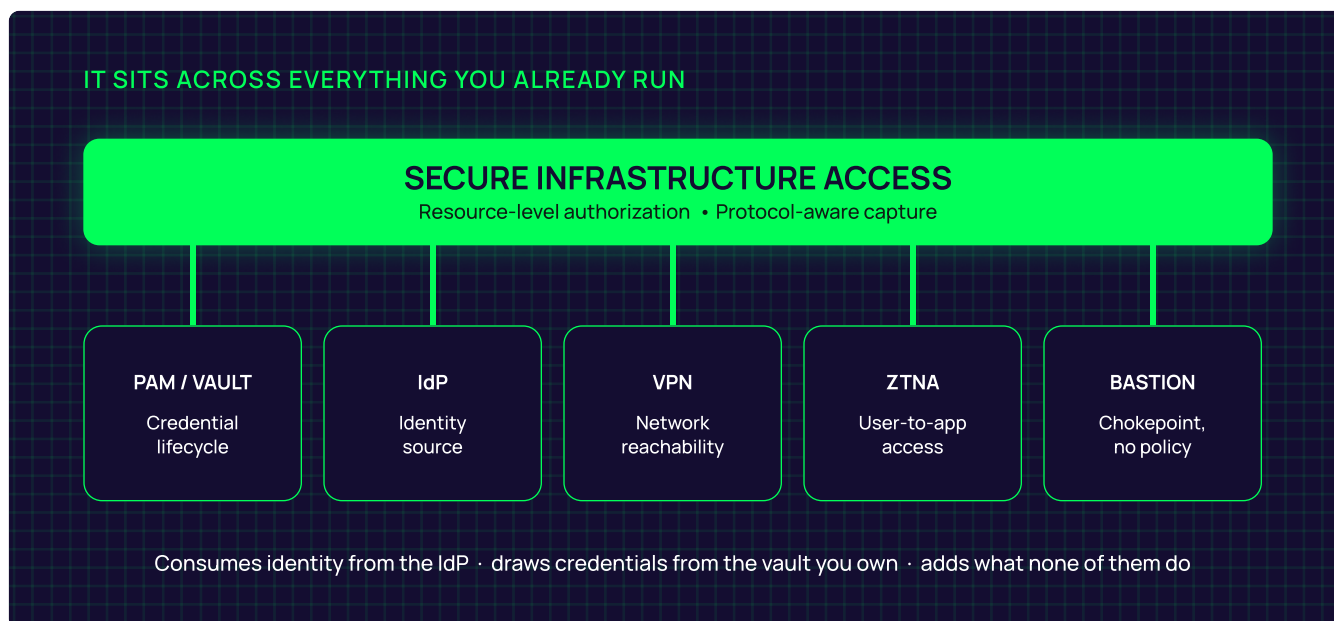
VPN. A VPN grants network reachability, and it answers the question of whether a device can route to a subnet. Once the packet lands, the VPN has no further opinion.

ZTNA. Zero trust network access replaces the VPN's implicit trust with per-application checks. The design targets user-to-application access, so interactive protocol access to a database or a cluster falls outside what the architecture was built to do.

Bastion hosts and jump servers. A bastion gives you a chokepoint, and concentrating traffic through a host you control helps, which is why so many teams have built one. A bastion does not evaluate policy, and it becomes both a target and a permanent maintenance obligation.

Cloud-native tooling. Session managers and identity-aware proxies govern access inside the provider that built them, and coverage ends at that provider's boundary by design. Multi-cloud, on-prem, and third-party database platforms fall outside it.

Infrastructure access sits across all of it. It consumes identity from the IdP, draws credentials from the vault you already own, and adds resource-level authorization plus protocol-aware capture for the systems engineers actually work in.



Why the category name keeps moving

You will see infrastructure access management, zero-trust infrastructure access, privileged access for DevOps, and dev-first PAM used for roughly the same set of capabilities. Analyst coverage splits across several categories, and none of them fit cleanly. Buy against the capability list in Step 3 and ignore the label.

What Breaks Without It

Standing credentials nobody rotates

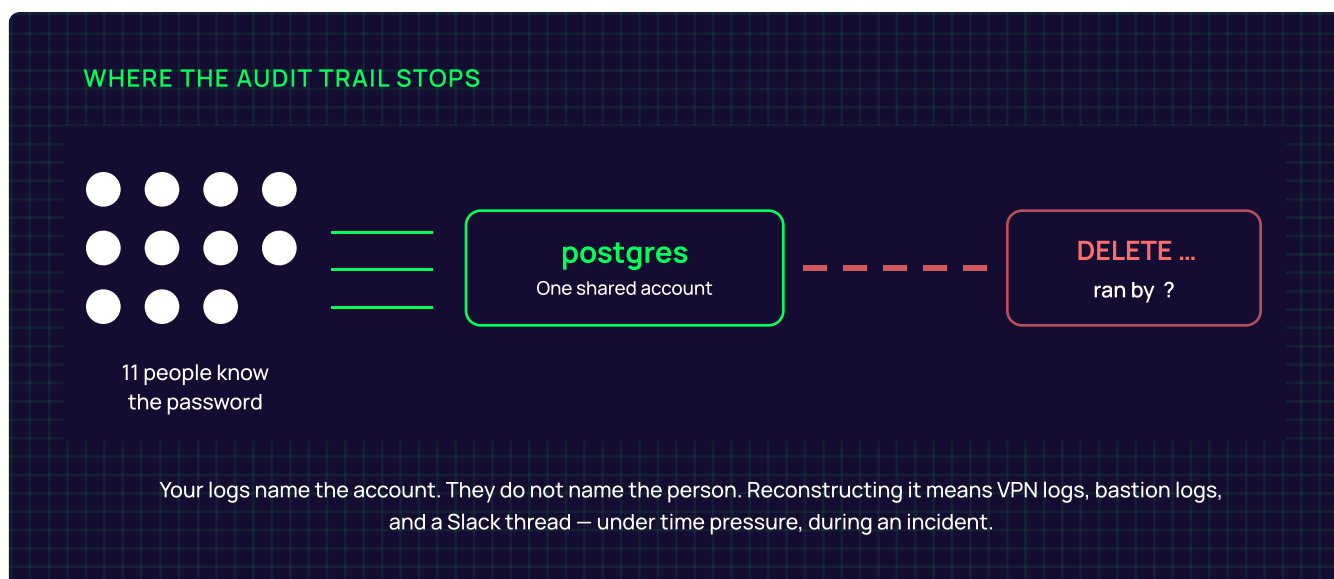
Every environment has them. The service account created for a 2021 data migration, the break-glass root password four people have memorized, the API key sitting in a Terraform state file. A rotation policy exists, and somebody defers rotation every time a rotation breaks something in production. Over a few years the count grows past the point where anyone can enumerate it, which is the real problem. You cannot revoke what you cannot list.

Shared accounts and an audit trail that stops at the account name

Your logs say the postgres account ran the DELETE, and they do not say which of the eleven people who know that password ran it. Reconstructing the answer means correlating VPN logs, bastion auth logs, and a Slack thread, under time pressure, during an incident. Most teams reach a probable answer, and few reach one they would defend in front of an auditor.

Network access when the requirement was resource access

A VPN connects a contractor to the network so they can access a single application server. It also puts them one Nmap away from everything else on that segment. Segmentation helps with this, and segmentation degrades every time somebody makes an emergency firewall change that nobody walks back later.



What Breaks Without It (continued)

Bastion hosts that stopped getting patched

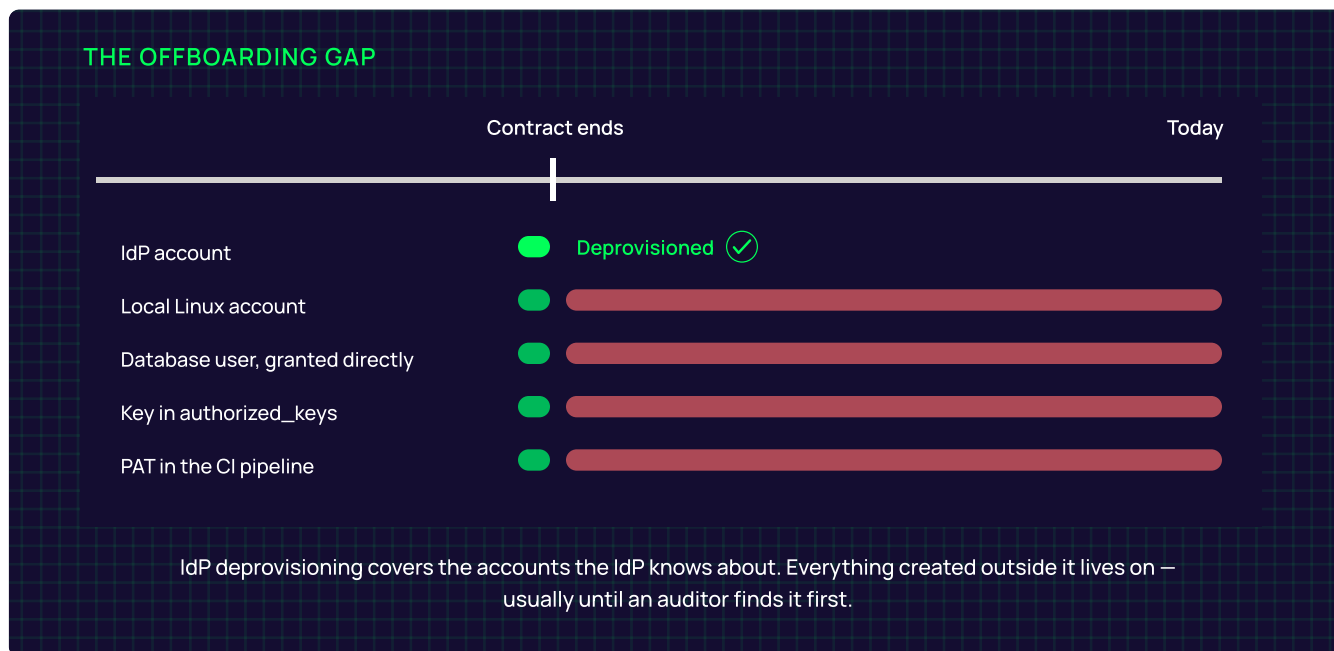
Somebody built the jump box, and that person left two reorgs ago. It runs an OS two majors behind, holds SSH keys for the entire fleet, and is quietly the most valuable target in the environment. Ask when it was last patched and watch the room.

Offboarding gaps

IdP deprovisioning works fine, and it covers the accounts the IdP knows about. It does not cover the local Linux account created during an outage, the database user granted directly, the public key appended to `authorized_keys`, or the personal access token wired into a CI pipeline. Contractor offboarding is worse, because it usually depends on somebody remembering. Service accounts have no offboarding process at all, which is why you still hold credentials for a vendor you stopped working with three years ago.

Audit evidence assembled by hand

The quarterly access review arrives. Someone exports IAM policies, pulls a vault report, screenshots group memberships, compiles it into a spreadsheet, and sends it to audit, and two engineers lose a week doing it. The result is accurate the day it lands and stale by the time anyone reviews it. Multiply that by SOX, PCI, SOC 2, and a customer security questionnaire, and a real share of your engineering capacity goes into producing paper.



AI Agents and Infrastructure Access

Agents change the shape of the access problem because they act with intent, at machine speed, and usually on behalf of a human who is not watching. They also reach production several ways at once, hitting databases, SSH hosts, Kubernetes clusters, and cloud consoles directly as well as through MCP servers.

A person requesting production database access creates one decision point. Someone approves it, the session happens, and the system records it in case anyone needs it later. Hand an agent that same access and you create thousands of decision points, most of them inside a window too short for a human approval workflow to reach.

Three assumptions in most access control models break here.

Approval workflows assume a pause. Just-in-time access built around a Slack approval works because humans tolerate a two-minute wait. An agent executing a remediation runbook does not have two minutes, and pre-approving the whole runbook leaves you with standing access under a different name.

Session boundaries assume a start and an end. Agent access tends to run persistently, and a video replay of a six-hour agent session is not an artifact any investigator will use.

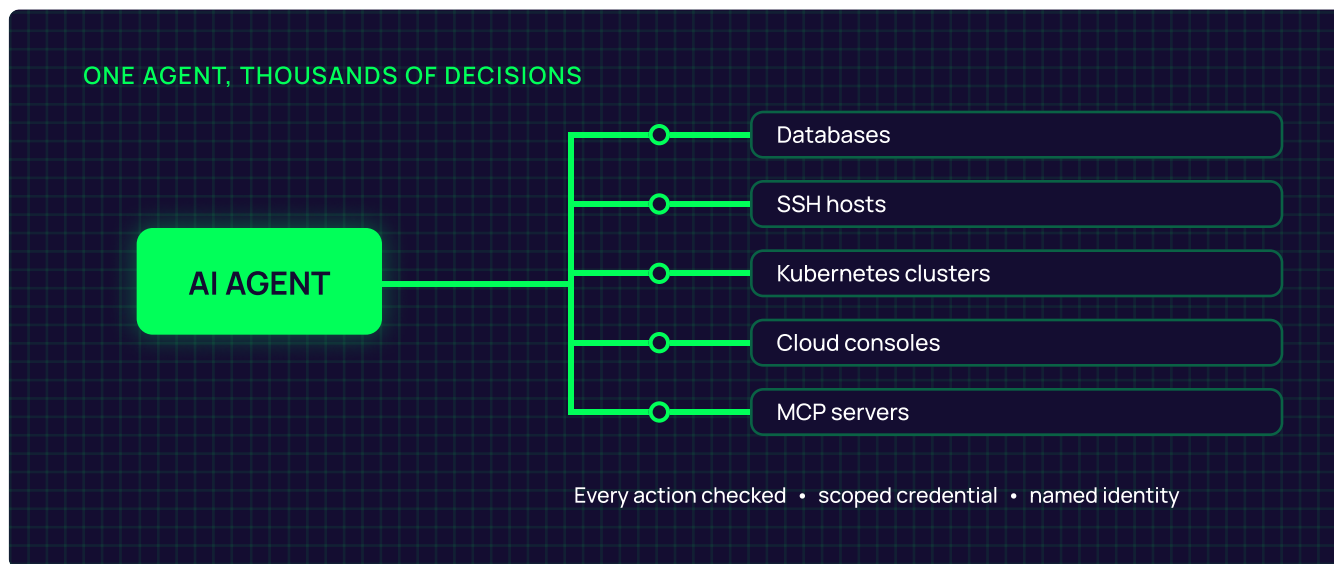
Attribution assumes a single actor. When an agent, invoked by a scheduler and running under a service account, acts on a ticket a human filed and drops a table, your log needs to capture all four facts. Most logs capture one.

Before an agent touches production, answer three questions. What can it reach, and how tightly is that scoped. Under whose authority does it act, and does that authority narrow to the task in front of it. What evidence do you keep, and can you reconstruct the decision the agent made rather than only the commands it ran.

Four practices are holding up so far:

- Give every agent its own identity rather than sharing one service account across several of them
- Scope each credential to a single task and expire it within minutes
- Evaluate authorization on every action rather than once per connection
- Build a kill switch that works at the access layer, so stopping an agent never depends on the agent cooperating

One caution. This part of the market is young, and claims across it are moving faster than documentation. Ask any vendor, Delinea included, for a live demonstration against your own use case rather than a roadmap slide.



What Is Driving Urgency Now

Infrastructure sprawl. The count of things requiring access grew faster than the process governing it. Cloud accounts multiply per team, clusters multiply per environment, and a managed database spins up in an afternoon. A platform team of twelve can be accountable for several thousand access-relevant resources, working from an access model designed for a few dozen.

Third-party access at scale. Contractors, MSPs, offshore teams, and vendor support engineers all need production access, and some of them need it for days rather than years. The onboarding path built for full-time employees does not fit them, so someone builds a parallel path that skips controls.

Service accounts outrunning governance. Non-human identities outnumber human ones in most environments by a wide margin. Somebody created each one for a reason, and almost none of them have a named owner or a review date.

Compliance and insurance. PCI DSS 4.0 tightened requirements around unique identification and multi-factor authentication for access into the cardholder data environment. SOX audits increasingly test privileged access to financially relevant systems rather than accepting a policy document as evidence. DORA and NIS2 put third-party and operational access under regulatory scrutiny across the EU. And cyber insurance renewals now ask direct questions about privileged access controls, with premium consequences for weak answers.

That last one moves budget more often than the others, so if you are building a case, start there.

FOUR FORCES, ONE THAT MOVES BUDGET

Infrastructure sprawl

Thousands of resources, a process built for dozens



Third-party access at scale

Days-long access on an onboarding path built for years



Service accounts outrun governance

Non-human identities outnumber humans, most unowned



Compliance & insurance

PCI DSS 4.0 · SOX · DORA · NIS2 · premium consequences



If you are building a case, start with the one at the bottom.

PART TWO

Defining What You Need

Step 1. Map Your Infrastructure

You cannot write an access policy for an estate you cannot list. Start with a one-page inventory. It does not need to be complete to be useful, and waiting for completeness is the most common way this project stalls in month one.

What it is and what it runs

Group by system of record rather than by hostname. "Production Postgres cluster, billing" tells you more than fourteen instance IDs ever will.

What happens when it is unavailable

Rank it on a four-point scale, running from revenue stops, to customers notice, to internal teams slow down, to nobody notices for a day. That ranking drives your phase order and how hard you argue about latency in Step 3.

Where it lives and who can reach it

Cloud, region, on-prem, colo. Flag anything with data residency constraints, because residency constrains deployment architecture and deployment architecture is expensive to change after you have signed.

In scope for phase one

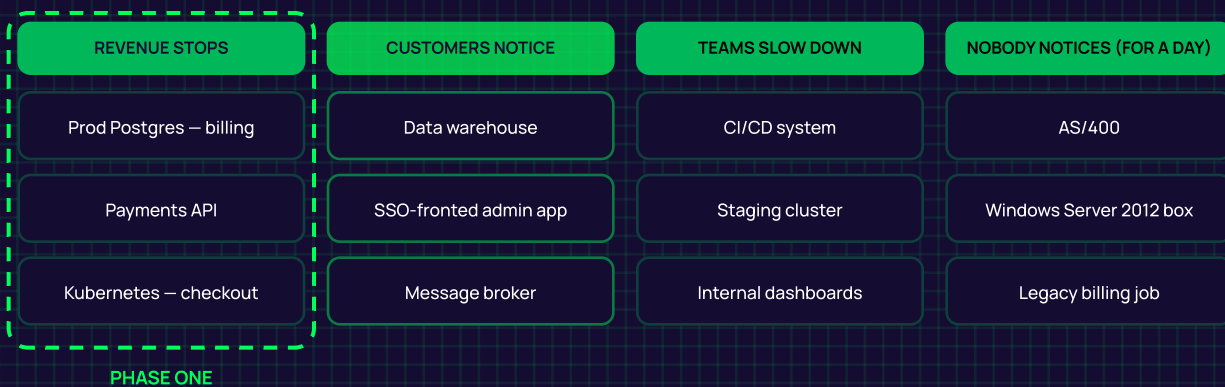
Be aggressive about saying no. A phase one covering one environment and three resource types that ships in eight weeks beats a comprehensive plan that ships never.

Add a short section on what is changing. A cloud migration in flight, an acquisition closing in six months, a database consolidation, a Kubernetes adoption that is 30 percent done. Anything on an 18-month horizon belongs in your requirements, because you are buying for the environment you will have.

Include your MCP servers in this inventory, and do not let them stand in for the whole agent surface. Agents reach databases, SSH hosts, clusters, and cloud consoles directly as well as through MCP, so an inventory that lists only MCP endpoints will miss most of what agents actually touch.

One more thing. Include the systems you are embarrassed about, whether that is the AS/400 or the Windows Server 2012 box running the process nobody understands. Those systems determine whether a vendor covers your estate, and a vendor will happily scope them out of a POC if you let them.

MAP THE ESTATE – RANK BY WHAT BREAKS WHEN IT'S DOWN



Group by system of record, not hostname · include the systems you are embarrassed about

Step 2. Map Everything That Needs Access

Same exercise, other side of the connection. People and automated processes go in the same inventory, because your policy engine must treat them the same way.

Employees. Break out by role and environment rather than by name. Backend engineers who need read access to production databases, SREs who need shell on everything during an incident, analysts who need the warehouse and nothing else. Note how many sit in each group and how often the group changes.

Contractors and third parties. Record the end date. Most environments have no field that stores a contractor's access expiry, which is precisely why contractor access outlives contracts. Record the internal sponsor too, since that is the person who approves and revokes.

Service accounts and machine identities. This part is unpleasant, so budget a week for it. Pull the list from every system that creates service accounts, which usually means the IdP, cloud IAM, the databases, and the CI system at minimum. Expect the count to surprise you. For each one, record what created it and whether anyone still owns it. A service account with no owner is a finding, and you want it in your own documentation before it lands in an auditor's.

FROM THE FIELD

We were publicly traded, and many of these accounts touched financial systems where SOX required full attribution for every action taken. We could not produce it. That is what started the inventory. We scoped North and South America, estimated 20,000 service accounts, and planned five weeks. The real number was over 100,000. The work took seven weeks. 52 percent had no owner. The worst single account we found held domain admin, permitted interactive login, had its password set to never expire, and that password was six and a half years old.

Treat your team's estimate as a floor. Ours was off by a factor of five, and that gap was the actual finding. It meant nothing in our process would have caught any of it, and nothing would have caught the next one either.

AI agents and what invokes them. Treat each agent as an identity in its own right. Record what triggers it, what it is permitted to reach, and whether it currently authenticates with a credential shared with a human process. If it does, write that down as a gap now.

For every entry, capture three facts. What it reaches today. How it authenticates today. Who approves a change to that access.

The gap between what an identity reaches and what its role requires is your scope of work, so quantify it. "Forty-one people hold standing production database access and eleven of them need it" is the single most useful sentence you can carry into a budget conversation.

Step 3. Define Your Requirements

A requirement is something that ends the evaluation if it is missing, and a preference is something you would like. Vendors are skilled at moving items from the first list to the second during a demo, so write both lists down before the first call and put a date on the document.

The test for a requirement is whether you can name the system or workflow that fails without it. “Must support PostgreSQL 12 because the billing database will not be upgraded before 2027” is a requirement. “Should have a modern interface” is a preference wearing a requirement’s clothes.

Eleven criteria follow. Take them into every demo in this order. Vendors will want to lead with their strongest capability, and a fixed order means you find each one’s weak spot in the same place every time.

1. Coverage

Protocol and resource coverage decides how much of your estate a vendor can actually front, and it is where scoping games happen. A vendor covering SSH, RDP, and PostgreSQL covers a lot of environments, and none of the interesting parts of yours if your estate includes Oracle, Kafka, and an internal admin app with its own authentication. Databases are where coverage claims usually thin out, so push past the engine list. Ask which engines get query-level capture rather than session video, which support policy on individual statements, and how managed services and read replicas behave differently from a self-hosted primary. Check every line of your Step 1 inventory against the supported list, check versions rather than product names, and ask the same question about MCP servers and the protocols your agents use directly.

ASK A VENDOR “Walk me through connecting to the ugliest system on my list. Show me live, do not describe it.”

2. Credential handling

The question is whether a credential ever reaches the human or process requesting access. Software that fetches a password from the vault and displays it has relocated the problem rather than removed it. Brokered access, where the software authenticates to the target on the requester’s behalf and the requester never holds the secret, removes an entire class of exposure. Ask how rotation works and what happens to an open session when a credential rotates underneath it.

ASK A VENDOR “Does the user ever see or hold the target credential? If yes, under exactly what circumstances?”

3. Policy model

Look at how the product expresses policy and how that holds up at scale. Role-based rules are simple, and they sprawl. Attribute-based rules scale better and get hard to reason about, so most real deployments run both. What matters more than the model is whether somebody who did not write a policy can read it six months later, and whether you can answer “what can this identity reach” without querying three systems.

ASK A VENDOR “Show me the policy granting a contractor read access to one database for two weeks. Now show me how I find every policy that touches that database.”

4. Just-in-time access and standing privilege

Eliminating standing access is the point of the exercise. Find out how somebody requests access, who approves it, how long it lasts, and what happens when the approver is asleep. Check whether the just-in-time path is fast enough for engineers to use during an incident, because if it is not, they will build a bypass and you will discover it two years later.

ASK A VENDOR “Across your existing customers, what is the median time from request to connected session?”

5. Runtime authorization

Connection-time checks decide whether an identity may open a session. Runtime authorization decides whether a specific action inside that session is permitted. The difference matters most for databases and agents, where a single approved PostgreSQL session can carry both a routine SELECT and a DROP TABLE. Find out whether the product can block the statement or only record it after the fact, and whether that enforcement holds across every engine you run rather than the one in the demo.

ASK A VENDOR “Can you prevent a specific SQL statement or shell command inside an already-approved session? Demonstrate it.”

6. Session capture and replay

Recording is table stakes, and useful recording is rarer than vendors imply. A video replay of a six-hour SSH session is evidence nobody will ever watch. What you want is structured capture, covering commands executed, queries run, and files transferred, each with a timestamp and an identity attached and searchable across every session. For databases this is the whole game. If a shared account fronts the connection, capture has to tie each statement back to the person or agent that issued it, or your audit trail stops exactly where the auditor starts asking. Ask about the storage footprint at your volume, because that number surprises people.

ASK A VENDOR “Find every session in the last 90 days where anyone ran a DELETE against the customers table. How long did that take you?”

7. Audit and evidence

The test is whether your auditor can pull evidence without your engineers assembling it. Ask which reports ship out of the box, whether an auditor can hold a read-only login scoped to their remit, and whether access reviews run inside the product or export to a spreadsheet the way they do today.

ASK A VENDOR “Can I give an external auditor a login that shows them everything they need and nothing else?”

8. Architecture and failure behavior

Deployment model determines latency, data residency, and what your bad day looks like. Understand where the control plane runs, where the data plane runs, whether session traffic transits vendor infrastructure, and what happens to access when the control plane is unreachable. Buyers skip that last question more than any other, and regret it more than any other.

ASK A VENDOR “Your control plane is down. Can my SREs still reach production? Walk me through the whole path.”

9. Break-glass

There will be a day when this software is part of the problem. Design the path around it before you need it, and make that path auditable, time-bound, and loud. Break-glass access nobody notices is standing access with a better name.

ASK A VENDOR “Show me the break-glass procedure, then show me the alert it fires.”

Step 3. Define Your Requirements (continued)

10. Integrations

This sits in the middle of a stack you already run. Identity comes from the IdP, credentials may come from the vault or PAM you already own, and logs go to the SIEM. Approvals may route through the ticketing system, and provisioning may need to hang off HR events. Every integration you cannot get natively becomes a script somebody on your team maintains forever.

ASK A VENDOR “Which of these are native, which run through an API I would call myself, and which would I be building from scratch?”

11. Administration and operating cost

Somebody runs this after the project ends. Find out how long onboarding a new resource takes, how you manage policy sprawl over time, and how many people comparable customers assign to it. If the answer is a fraction of one person, ask to speak with a reference at your scale.

ASK A VENDOR “How many FTEs do customers our size dedicate to running this day to day?”

THE TEST FOR A REQUIREMENT

REQUIREMENT

“Must support PostgreSQL 12 – the billing database will not be upgraded before 2027.”

- ✓ Names the system that fails without it
- ✓ A zero here ends the evaluation
- ✓ Written down before the first call

PREFERENCE IN DISGUISE

“Should have a modern interface.”

- ✗ Names nothing that fails
- ✗ Vendors love this list
- ✗ Moves down during every demo

Write both lists before the first call – and put a date on the document.

Step 4. Build the Business Case

Evaluations die in procurement more often than they die in the POC, and the fix is starting the cost model before the technical evaluation ends.

The committee. Five people can stop this, each for a different reason. The economic buyer wants the number and what it replaces. Security architecture wants assurance this does not become another silo, and platform ownership wants to know it will not slow engineers down. Compliance wants the evidence story in writing, and internal audit wants to know the control is testable. Meet each of them before the POC rather than after, because a surprise objection in week ten costs a quarter.

What to count. Start with numbers you can defend:

- Licensing and infrastructure cost for the software itself
- Administrative hours spent today on access provisioning and deprovisioning, measured over two weeks rather than estimated
- Engineering hours per audit cycle, multiplied by audits per year
- Time from new engineer start date to first productive access
- Tooling you retire, including bastion infrastructure and any per-seat cost on tools this displaces

Then the softer items, framed carefully. Reduced investigation time during incidents, faster contractor onboarding, and a lower cyber insurance premium if your broker will commit to a number. Some brokers will, and it is worth asking.

What not to claim. Skip breach-cost-avoidance math. Multiplying an industry-average breach cost by a probability you invented does not survive contact with a CFO, and using it damages your credibility for the parts of the model that are real. If you want to talk about risk, talk about audit findings you can close and controls you can evidence, because those are verifiable, and verifiable is what gets funded.

FROM THE FIELD

The number that moved our budget was not a risk score. It was the count from that inventory, put in front of the people holding the money. More than 100,000 privileged credentials in scope, and an owner we could name for fewer than half.

Nobody argued with it. There was nothing to argue with. It was a count rather than a projection, and everyone in the room understood that the same question was going to arrive from an auditor.

Probability estimates invite debate. A basic question about your own environment that you cannot answer does not. If you build one artifact before your funding conversation, build the list.

PART THREE

Evaluating, Selecting, and Deploying

Step 5. Understand the Approaches

Six architectural approaches cover the market. Sort vendors by these rather than by tiers, because architecture determines what a product can do, and tiers just tell you who spent more on marketing last year.

Bastion hosts and jump servers

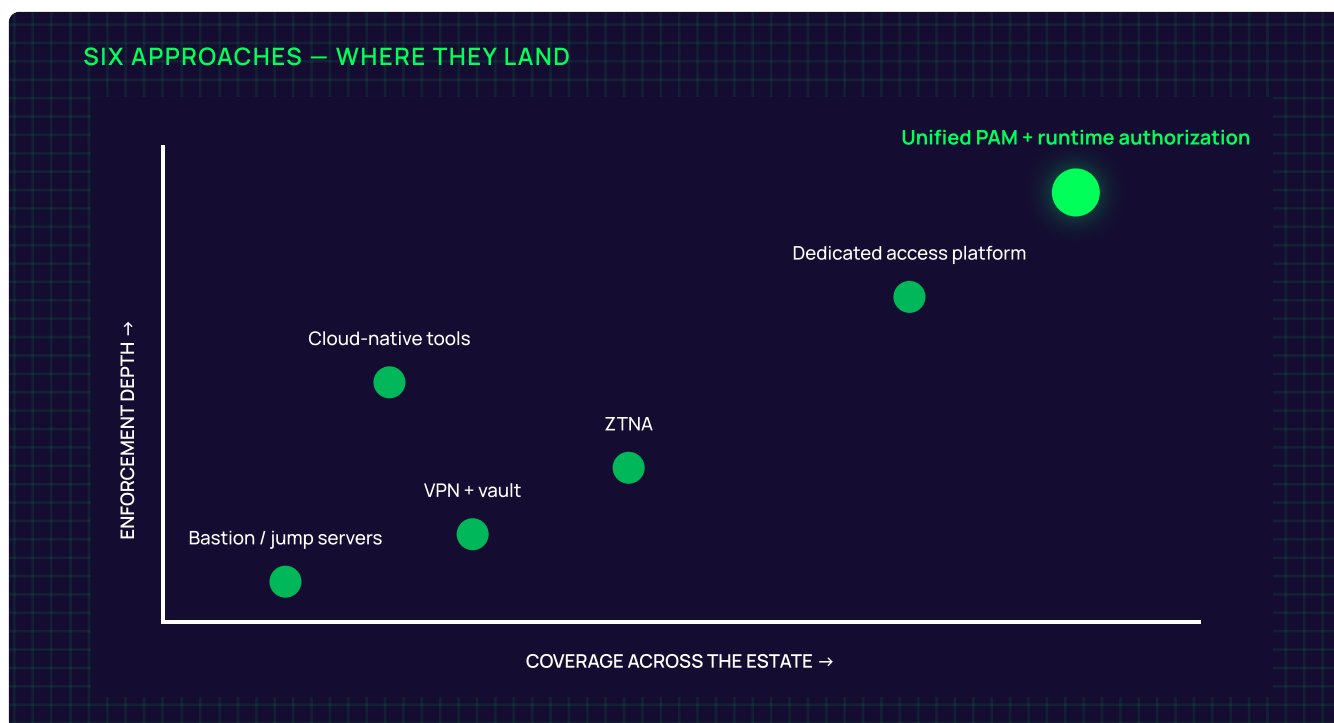
Cheap, familiar, and they give you one chokepoint for traffic and one place to collect logs. They carry no policy engine, no per-resource authorization, and no credential brokering. The bastion becomes a high-value target and a permanent patching obligation, and it does nothing for database or Kubernetes access. Low license cost, high and permanent operational cost.

VPN paired with a credential vault

This combination covers network reachability and credential hygiene, and most companies already own both halves. The halves do not know about each other. The VPN grants network access without knowing what the user checked out of the vault, and the vault releases a credential without knowing where the user is connecting from, so nothing correlates them into a single session record. The cost is usually already sunk, which makes it attractive and makes the seams easy to ignore.

Cloud-native session managers and identity-aware proxies

Deep integration with the cloud that built them, no extra infrastructure, and attractive pricing. These are designed to govern access within a single provider's ecosystem, and coverage ends at that boundary by design. Run multi-cloud or on-prem, or run a database platform outside the provider's ecosystem, and you end up operating two or three of these and reconciling their logs by hand. Low direct cost, real cost in fragmentation.



Step 5. Understand the Approaches (continued)

ZTNA and network-layer access platforms

Good replacement for VPN trust, strong user experience for web applications, solid device posture integration. The architecture was built for user-to-application access, so protocol-level access to databases and clusters falls outside its design intent, and session capture generally operates at the network layer. Moderate cost, and it may already be in your stack for the workforce use case.

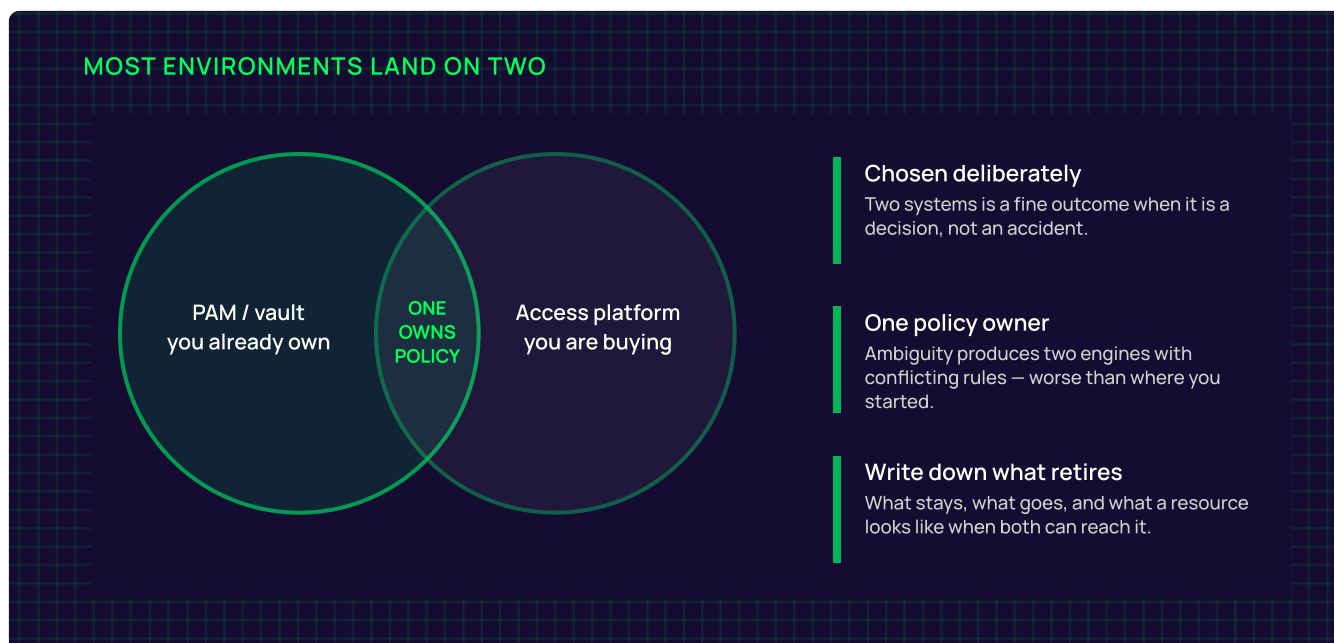
Dedicated infrastructure access platforms

Protocol-aware access across servers, databases, clusters, and cloud, under one policy model, with credential brokering and capture at the command and query level. It is another system to run, and it needs to integrate cleanly with the vault and IdP you already own. Coverage for legacy and less common systems varies a lot by vendor, which is why Step 3 starts with coverage. Direct cost runs above the point solutions, so compare it against the total of what it lets you retire rather than against any single line item.

Unified PAM with runtime authorization

Enterprise privileged access management and per-action authorization under one policy model and one audit record, spanning human, machine, and agent identities. Credential lifecycle, brokered access, session capture, and authorization of the action itself sit in the same place, which removes the correlation work that the other five approaches leave you holding. This is the newest of the six and the one with the most vendor movement right now, so verify what a vendor has shipped rather than what it has announced. The trade here is consolidation, and the evaluation should turn on coverage breadth and on whether the product evaluates authorization per action rather than per connection.

Most environments land on two of these rather than one. That is a fine outcome as long as you chose it deliberately and you know which one owns policy.



Step 6. Run the Proof of Concept

Write your own use cases. The vendor will offer a POC plan. Read it, then replace it. Pull five to eight use cases straight out of your Step 1 and Step 2 inventories and make at least two of them ugly, covering the legacy database, the contractor who needs access for nine days, and the 3am incident. A vendor-authored POC plan is a demonstration, and you already sat through the demonstration.

Set thresholds before the first call. Write down what a pass looks like, with numbers wherever you can. Session establishment under two seconds. A new resource onboarded in under thirty minutes by someone who does not work for the vendor. An auditor pulling last quarter's privileged sessions with no engineering help. Vague criteria produce a scoring meeting where whoever speaks last wins.

Get the right people in the room.

- The analysts and engineers who will use it daily, because adoption lives or dies in their hands
- A DBA, who will find the connection pooling problem nobody else would have found
- A DevOps or SRE lead, who will tell you whether this survives an incident
- Somebody from compliance or internal audit, who will tell you whether the evidence is usable
- An executive sponsor, who unblocks procurement and who needs to have watched at least one live session

The DBA is the person most often left out and the person most likely to surface a blocking issue in week two instead of month six.

Test the databases properly. Database access is where these evaluations break, and it is rarely the part anyone plans for. Run a connection through your pooler and confirm nothing downstream loses the client identity. Check that query-level capture attributes each statement to a named person even when a shared account fronts the connection. Try to block one statement inside an already-approved session. Point the tool at a managed instance and at a read replica, because both often behave differently from a self-hosted primary. Then connect a service account and an agent to the same database, since that is the path nobody tests and the one that turns up in the audit finding.

Timebox it and be honest about the limits. Two weeks proves connectivity, user experience, and basic policy authoring. It does not prove operating cost, policy sprawl at scale, or upgrade behavior, so for those use customer references at your size and ask pointed questions. Stretching a POC to six weeks will not answer year-two questions, and the extension mostly just builds sunk-cost pressure to buy.

Score against the checklist. Use Appendix B, same criteria in the same order for every vendor, filled in during the session rather than from memory afterward. Impressions favor whoever demoed most recently.

Test the failure paths. Pull the network, and break the control plane connection if the vendor allows it. Misconfigure a policy on purpose and find out whether it fails open or closed. Run break-glass end to end and confirm it fired an alert. A POC that only exercises the happy path tells you what year one looks like and nothing about the day something goes wrong.

TWO WEEKS, TIMEBOXED

Day 1



Your use cases,
not the vendor's

Day 3



Ugly systems
connected live

Day 6



DBA tests pooling
& attribution

Day 9



Failure paths &
break-glass

Day 10



Scored against
Appendix B

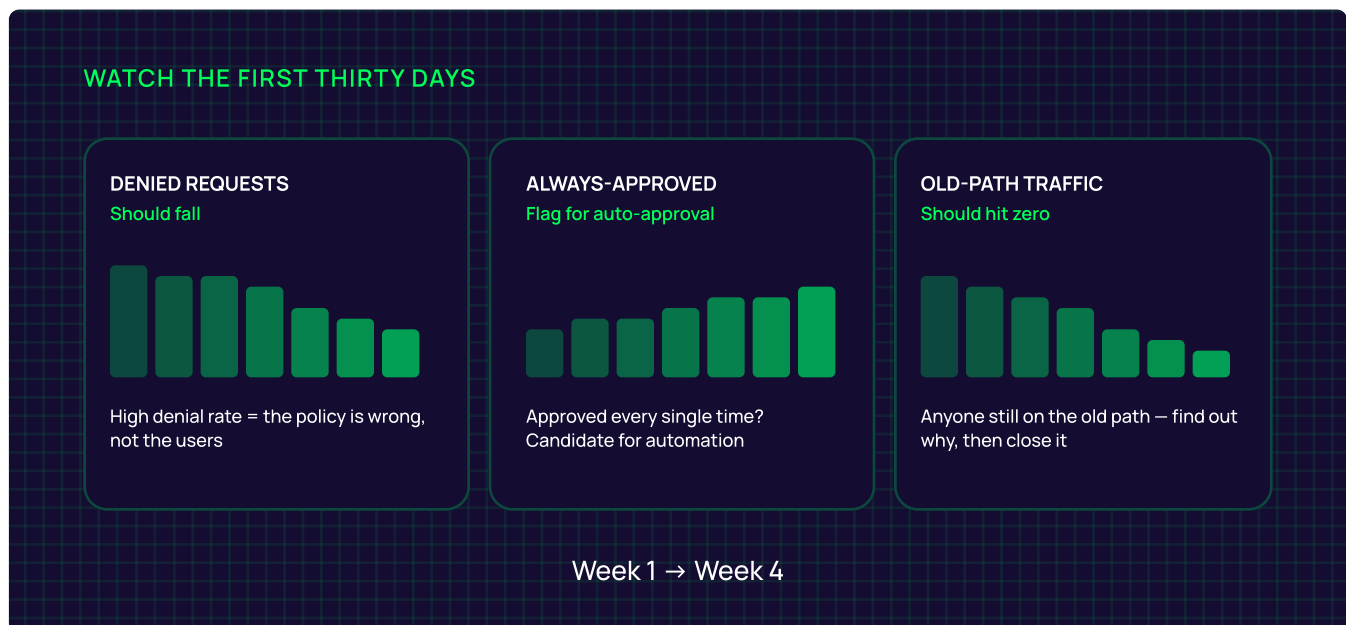
Step 7. Deploy

Sequence by friction. Start with the group that hates the current process most. They will tolerate rough edges because the alternative is worse, and their enthusiasm carries the next group. Starting with the most critical environment is defensible on paper and slow in practice.

Plan coexistence explicitly. Most buyers are adding this to a stack rather than replacing one, so decide up front which system owns which job. The IdP stays the source of identity. The vault or PAM platform retains ownership of the credential lifecycle for the accounts it already manages, and the access tooling consumes from it rather than duplicating it. Write down what gets retired, what stays, and what a resource looks like when both systems can reach it. Ambiguity here produces two policy engines with conflicting rules, which is worse than where you started.

Prepare users for the part that gets slower. Some workflows will feel worse before they feel better, and an engineer who used to type `ssh prod-db-01` from memory now goes through a request. Say that plainly in the rollout communication, explain the trade, and hand them the fastest path you have. Surprise friction produces workarounds, and a workaround you discover in month eight undermines the entire control.

Watch the first thirty days. Track denied requests, since a high denial rate usually means the policy is wrong rather than the users are. Track requests approved every single time, because those are candidates for automatic approval. Track anyone still connecting through the old path and find out why before you close it.



Step 8. Reach Steady State

Set your baseline before deployment or you will not be able to prove any of this later. The measurement takes an afternoon during Step 2 and returns ten times the effort.

Treat the numbers below as recommended starting targets to agree with your own stakeholders rather than industry benchmarks. Measure your baseline first, then negotiate the number you will be held to.

Metric	What it tells you	Recommended starting target
Percentage of in-scope infrastructure behind the control point	Coverage, and whether phase two stalled	90% of phase one scope by month 6
Percentage of standing privileged access eliminated	Whether the core risk actually moved	80% for human access against your Step 2 baseline
Median time from request to connected session	Whether people will use the sanctioned path	Under 5 minutes routine, under 60 seconds pre-approved
Time from new engineer start to first productive access	Onboarding cost, and an easy number for the business case	Same day
Engineering hours per audit cycle	Whether the evidence promise held up	80% reduction against baseline
Access review completion rate and cycle time	Whether governance is real	100% completion, closed within the quarter
Percentage of service accounts with a named owner	The one everyone skips	100%, and hold it there

Step 8. Reach Steady State (continued)

Reviews. Run quarterly access reviews inside the tool rather than in a spreadsheet. Assign the reviewer to the person who owns the resource rather than the user's manager, because the resource owner knows whether the access makes sense. Managers approve everything.

Expansion. Phase two is usually the second and third environments. Phase three is machine identities, which is harder because ownership is unclear. Phase four is agents. Resist doing these in reverse order however urgent the agent problem feels, because the policy model you build for humans is the one your agents will inherit.

FROM THE FIELD

Once we had the list, the instinct was to start disabling accounts. That is how you cause an outage and lose the mandate.

What worked was slower. We mapped which resources and services each account actually touched, then prioritized by the tier of the asset behind it. Temporary suspension first, with a fast path to restore, rather than deletion. If nothing broke, the account stayed suspended. If something broke, we finally had the owner we had been looking for, because someone always calls.

Breaking a production process to prove a point sets the program back a year. Suspension with a rollback path gets you the same information and keeps the room on your side.

Warning signs. Adoption has stalled if the old access path still carries traffic after month three, if teams onboard new resources outside the tool, if the denial rate never falls, or if administration has quietly consolidated onto one person who is now a single point of failure.

ADOPTION HAS STALLED IF...



Old path still carries traffic
After month three



Resources onboarded outside the tool
New systems skip it



Denial rate never falls
The policy is fighting the users



Admin consolidated onto one person
A single point of failure

PART FOUR

How Delinea Approaches This

Everything above holds regardless of what you buy. Here is where Delinea lands against it.

Delinea completed its acquisition of StrongDM on March 5, 2026, and enterprise PAM and just-in-time runtime authorization now run as one platform rather than two products. That matters for this guide because of Step 3. The criterion that separates working products from demonstrations is whether authorization happens on the action or only on the connection, and that is the criterion the combined platform exists to answer.

One policy and one audit record across every identity. Privileged access management and runtime authorization sit on the Delinea Platform, powered by Delinea Iris AI. Employees, contractors, service accounts, and AI agents answer to the same policy model and land in the same audit record. Nobody correlates logs across two products to answer what an identity reached.

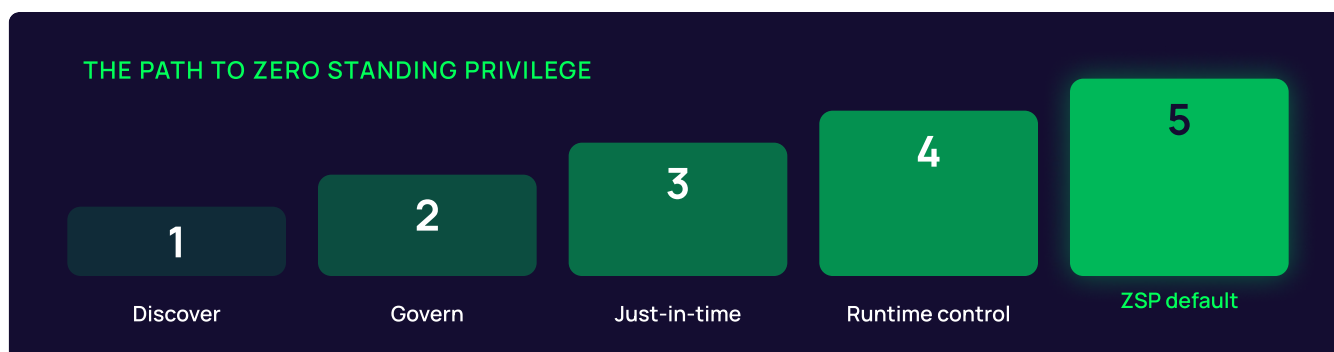
No credential reaches the requester. Access runs through the control plane rather than getting granted directly. The platform injects the credential at the moment of use, scopes it to the task rather than inheriting it from the person who deployed the agent, and revokes it when the task finishes.

Policy enforced on every action, not only when the session opens. A single AI session can carry dozens of tool calls, each with a different risk profile. The platform evaluates each one on its own and allows it, blocks it, or pulls a human in before it executes. Delinea released these runtime authorization capabilities for AI agents in July 2026, and they run on the existing platform without new infrastructure.

Coverage that does not depend on how the agent connects. Agents reach production databases, SSH hosts, Kubernetes clusters, cloud consoles, and MCP servers, and they frequently reach them directly rather than through an MCP server. The platform authorizes and records each of those connections through a single control point across every protocol agents use. Anything that sees only the traffic routed through its own gateway leaves the direct connections outside its view.

Agent-aware policy and attribution that survives an investigation. The platform identifies whether a connection is agent-driven or human-driven before it grants access, and applies agent-specific policy at that moment. Every tool call, database query, and SSH command ties back to a named identity.

Zero standing privilege as a sequence rather than a switch. Delinea runs ZSP across five stages. Discover identities and their access, govern accounts and secrets, make access just-in-time, control access at runtime, and eliminate standing privilege as the default. Programs that stop after the first three have done the visible work and left the exposure that matters. Existing PAM investments carry forward, and none of this requires rip-and-replace.



APPENDIX A

Vendor Question Bank

Bring these to every vendor call, grouped the same way as the Step 3 criteria.

Coverage

- Which of these exact systems and versions do you support natively today?
- Show me a live connection to our oldest system.
- What happens to a resource type you do not support natively?
- How long does it take to add support for something new, and who pays for that?

Credential handling

- Does the requester ever see or hold the target credential?
- Where do credentials live, and can you use the vault we already own?
- What happens to an open session when the underlying credential rotates?
- How do you handle systems that cannot support rotation?

Database access

- Which database engines and versions do you support, and which of those get query-level capture rather than session video?
- When a shared database account fronts the connection, does capture still attribute each statement to a named identity?
- Can you block or require approval for a single statement inside an open session, and does that hold for PostgreSQL, MySQL, SQL Server, Oracle, and MongoDB alike?
- What happens to capture and enforcement when we put a connection pooler in front of the database?
- How do managed database services and read replicas differ from a self-hosted primary in your product?
- What latency do you add to a query, measured, at our connection volume?
- How does a service account or an AI agent connect to a database through you, and what does the audit record look like afterward?

Policy model

- Show me the policy language. Who on my team would author it?
- How do I answer “what can this identity reach” in one query?
- How do I find every policy touching a given resource?
- What prevents policy sprawl after two years?
- How do policy changes get reviewed and rolled back?

Just-in-time and standing access

- What is the median time from request to connected session across your customers?
- How does approval work when the approver is unavailable?
- Can access expire automatically? What is the shortest duration you support?
- What standing access remains after a full deployment?

Appendix A. Vendor Question Bank (continued)

Runtime authorization

- Can you block a specific command or query inside an approved session?
- Is that enforcement or alerting?
- How does that work for a database versus a shell versus a cluster?
- What is the latency cost of inline enforcement?

Session capture

- What exactly gets captured for each protocol?
- Is capture searchable across sessions, and how fast at our volume?
- What is the storage footprint per thousand sessions?
- Who can view a recording, and is viewing itself logged?

Audit and evidence

- Which reports ship out of the box?
- Can an external auditor have a scoped read-only login?
- Do access reviews run inside your product?
- Which SOX and PCI DSS 4.0 control objectives, and which SOC 2 Trust Services Criteria, does this evidence map to?

Architecture

- Where does the control plane run and where does the data plane run?
- Does session traffic transit your infrastructure?
- What are our data residency options?
- What is the added latency, measured, at our geography?
- What is your SLA and what does the last twelve months of uptime look like?

Failure and break-glass

- Your control plane is unreachable. Can we still get to production?
- What is the break-glass procedure, and what alerts does it fire?
- Does a policy misconfiguration fail open or closed?
- What is the upgrade process and does it require a maintenance window?

Integrations

- Which of our systems do you integrate with natively today?
- Which integrations are API-only and which would we build ourselves?
- How does provisioning tie to HR or IdP lifecycle events?
- What does the SIEM integration actually send?

Administration and cost

- How many FTEs do customers our size run this with?
- How long does onboarding a new resource take, done by us?
- What does the pricing model do when our resource count doubles?
- What is included in support, and what costs extra?
- Give me two references at our scale, in our industry, running for more than a year.

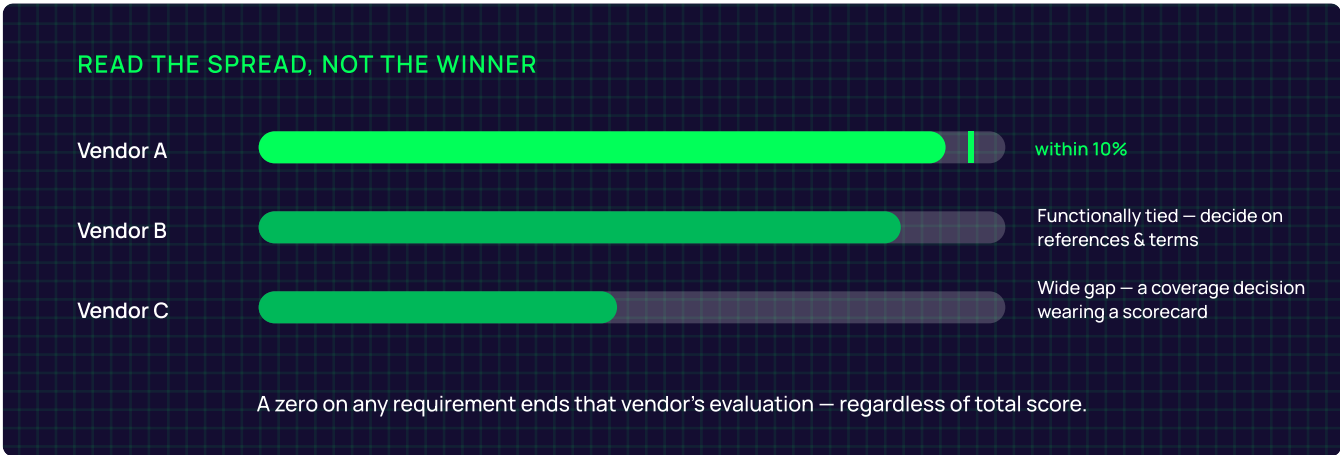
APPENDIX B

Requirements Checklist and Scoring Template

Mark each criterion as a requirement or a preference before the first vendor call, and weight each one from 1 to 5. Score each vendor from 0 to 5 during the session rather than afterward. A zero on any item you marked as a requirement ends that vendor’s evaluation regardless of total score.

Criterion	Req / Pref	Weight	Vendor A	Vendor B	Vendor C	Notes
Coverage of our estate						
Credential handling and brokering						
Policy model and readability						
Just-in-time access and expiry						
Runtime authorization						
Session capture and search						
Audit and evidence self-service						
Architecture, latency, residency						
Failure behavior and break-glass						
Integrations with our stack						
Administration and operating cost						

Total the weighted scores, then look at the spread rather than the winner. Two vendors within ten percent of each other are functionally tied, and the decision should move to references, roadmap fit, and commercial terms. A wide spread usually means one vendor covers your estate and the other does not, which is a coverage decision wearing a scorecard.



APPENDIX C

How Evaluations Go Wrong

The POC tested one protocol. SSH worked beautifully, and nobody tested the Oracle database, which is where the auditor is going to look.

The vendor wrote the use cases. You evaluated the product they wanted to show rather than the environment you have to run.

Nobody invited the DBA. The connection pooling problem then surfaces in month six, in production, during a peak.

The team tested only the happy path. Nobody pulled the network cable, nobody misconfigured a policy on purpose, and nobody ran break-glass.

The pilot group was five friendly engineers. Five people who volunteered will tolerate anything, and fifty who did not volunteer will not.

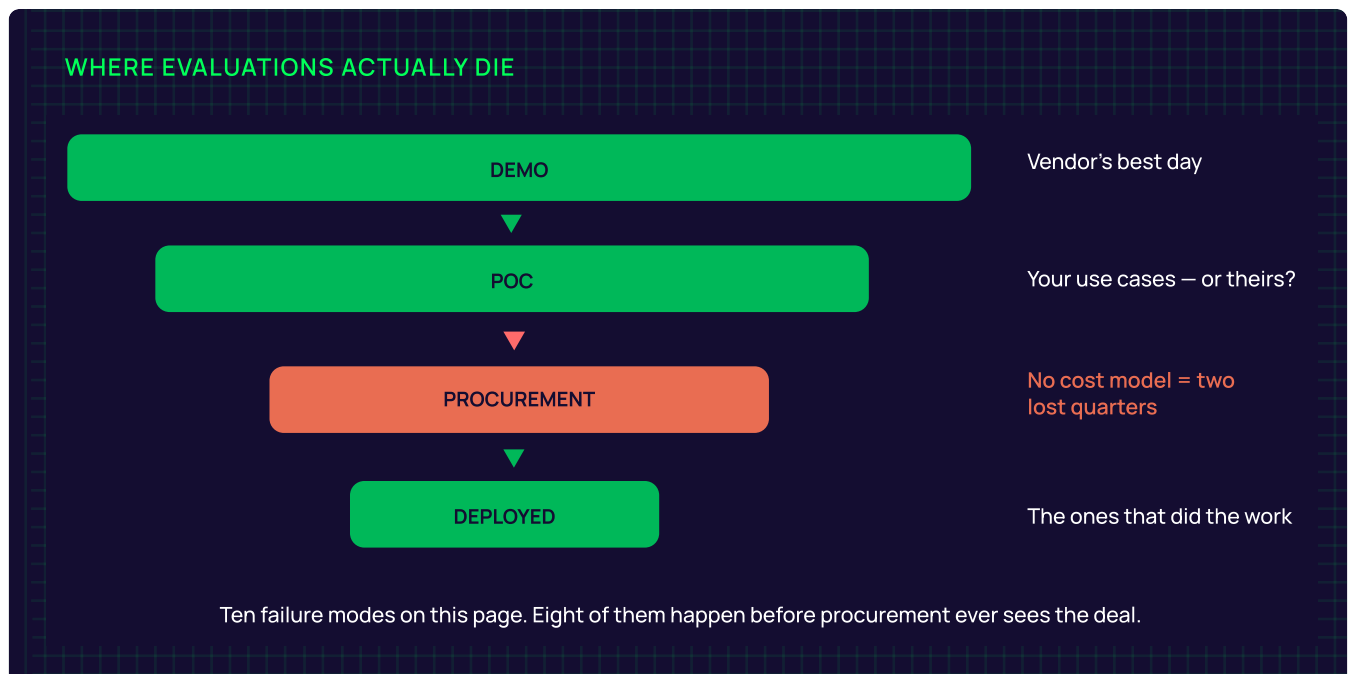
Nobody built the cost model. The technical evaluation finished in week eight and the deal sat in procurement until the next fiscal year.

Nobody pinned down coexistence. Two policy engines went live with overlapping rules and no clear owner, and the first conflict took a week to untangle.

The just-in-time path was slower than the bypass. Engineers built a workaround during the first incident, and it quietly became the normal path.

Somebody wrote the success criteria after the demos. The scoring meeting turned into an argument about impressions, and the vendor who demoed last won.

Nobody measured the baseline. Twelve months later everything works and there is no way to prove it.



APPENDIX D

Glossary

Just-in-time access. Access that lasts only as long as a specific task and then expires automatically, rather than persisting.

Zero standing privilege. A state in which no identity holds persistent elevated access to a resource, and every elevation happens per request.

Credential brokering. The access software authenticates to the target system on the requester's behalf, so the requester never holds the credential.

Runtime authorization. Policy evaluated on each action inside a session rather than only when the session opens.

Non-human identity. A service account, workload identity, API client, or AI agent that authenticates and acts without a person driving each request.

Identity-aware proxy. A gateway that authenticates the identity behind a request and applies policy before forwarding traffic to the target.

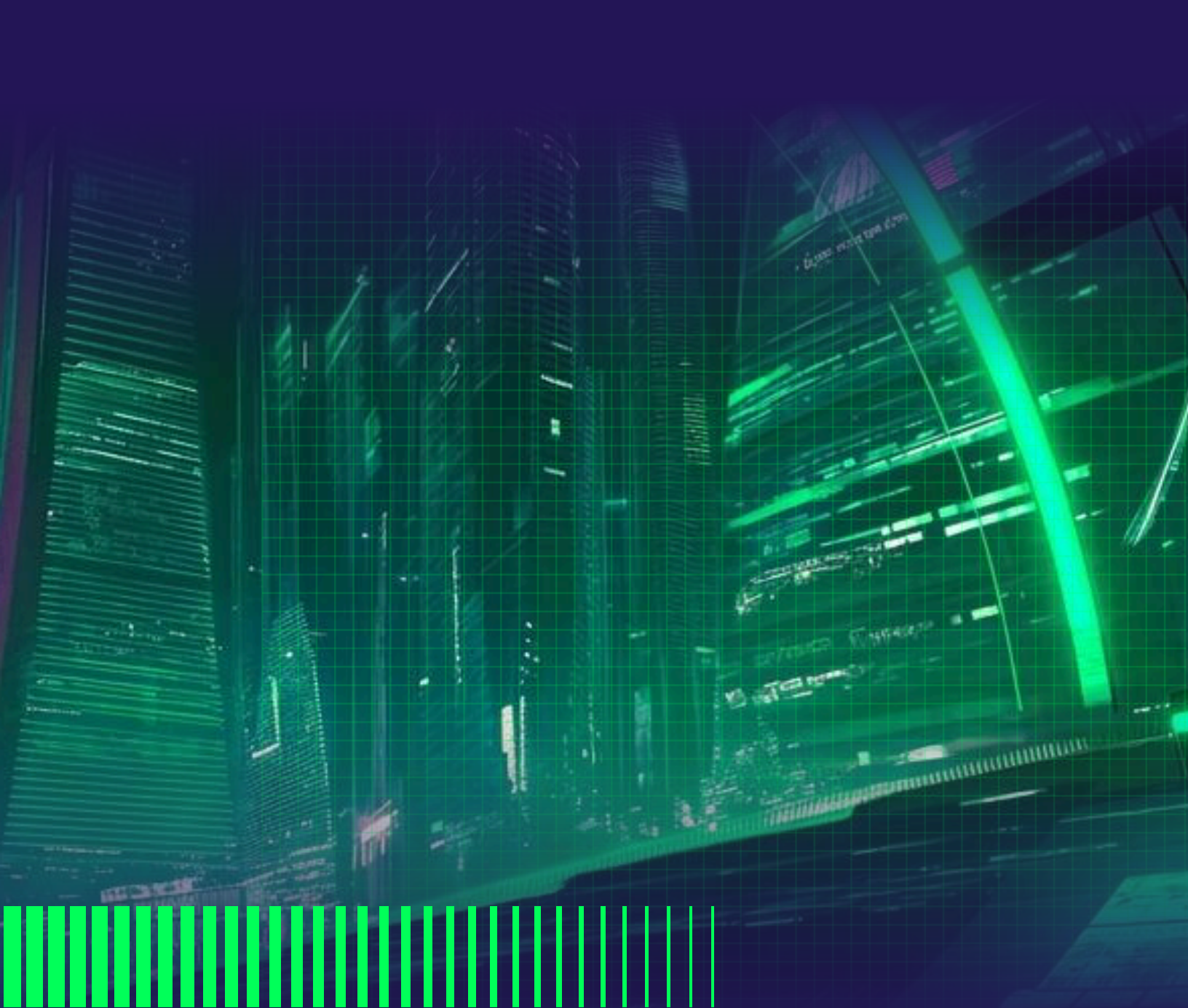
Session capture. Structured recording of what happened during a session, including commands, queries, and file transfers, attributed to an identity.

Break-glass. An emergency access path you design in advance for the times normal access controls are unavailable. It should be time-bound, alerted, and audited.

HOW THE TERMS CHAIN TOGETHER



Every request flows through the top row. Every term on the bottom row exists to get you to the one on the right.



Delinea

Delinea is the identity security platform that governs what humans, machines, and AI agents do once they have access. It enforces just-in-time authorization at the moment of action, not just when access is first granted, reducing reliance on standing privilege across on-premises, multi-cloud, and ephemeral infrastructure. Built on deep privileged access management expertise, the Delinea Platform continuously discovers identities, analyzes risk, and enforces least privilege, so every access decision is defensible. This gives organizations the visibility and control to adopt AI with confidence. Learn more at delinea.com.