sonraí security

SONRAI SECURITY | WHITEPAPER

Just-in-Time Access for Humans

How the Sonrai Cloud Permissions Firewall eliminates standing human privilege — without an entitlement cleanup project, and without interrupting the people doing the work

Human access to cloud accounts has been governed the same way for years: grant a standing permission set, recertify it in a quarterly access review, and hope the review catches what changed. The result is predictable: administrators and on-call engineers carry powerful permission sets around the clock, access reviews rubber-stamp what nobody remembers granting, and a single phished credential inherits everything its owner could ever do. The alternatives haven't fared better - PAM vaults translate poorly to cloud-native access, and CIEM remediation hand-trims entitlements one ticket at a time.

Just-in-Time (JIT) access in the Sonrai Cloud Permissions Firewall takes standing privilege off the table without a cleanup project. Humans request access to a permission set in a specific account when they need it, through Slack or Microsoft Teams; the grant is time-bound and expires automatically. Permission sets never sanctioned for an account can be denied outright - even ones never enrolled in JIT. Dormant identities are quarantined rather than deleted, and every session ends with an AI-generated summary. Privileged access exists only while it is being used, is approved by the people with context, and leaves a reviewable record every time.

Standing Privilege and the Access-Review Treadmill

The core defect of standing access is temporal: a permission set granted for a task in March is still live on a November weekend for an attacker who has compromised the credential. Quarterly recertification of hundreds of assignments degenerates into bulk approval — the reviewer cannot know what is still needed, so the safe answer is to keep it. Trimming entitlements runs into the same wall as every ticket-driven program — each removal is a negotiation, each risks blocking legitimate work, and the estate regrows faster than it is pruned. What is needed is a model where cleanup is unnecessary.

Access That Exists Only When It Is Needed

Permission sets are enrolled in JIT and scoped to the accounts where they are sanctioned. A user makes a request - from Slack, Teams, or just logs into the AWS console - naming the permission set, the account, and the duration. It routes to the scope's owner, who approves where they already work. When the grant expires, the access is simply gone: nothing to revoke, nothing to recertify, nothing for an attacker to find. Unanswered requests escalate to the approver at the next level up the scope hierarchy; unapproved requests are denied. A default deny state persists otherwise.

No prior request flow is required. A user can log in exactly as they did before JIT existed: the AWS console returns an access error, and within moments a Slack message from the Sonrai app asks them to justify the need (or self-approve where allowed). Even a user who missed the JIT training is never dead-ended; the request flow comes to them via Slack or Teams. An optional browser extension makes the same experience seamless right in the console.

> **Privileged access exists only while it is being used, is approved by the people with context, and leaves a reviewable record every time.**

No cleanup of already-existing entitlements

Adopting JIT doesn't start with an audit of who has what. Existing permission sets, assignments, and IAM policies stay exactly where they are — no rewriting, no revocation campaign. Enforcement sits above existing entitlements: what is approved for an account works, what is not is denied, and the standing grants underneath simply stop mattering. Organizations reach a JIT posture in days.

Optional deny-first: unsanctioned permission sets are blocked, enrolled or not

Most JIT products only govern what has been enrolled in them. To ignore access that is not enrolled in JIT — and therefore under-managed and forgotten — is to ignore the riskiest access. The Cloud Permissions Firewall can invert this: it blocks permission sets not approved for an account, whether or not they were ever enrolled in JIT. Assignments created years ago, provisioned too broadly, or added out of band are unusable where they haven't been sanctioned.

Zombie identities: quarantined safely, revived through JIT

Every organization carries identities that stopped being used long ago - a test workload, an abandoned AI hackathon project, a former contractor, the admin user from a migration that finished two years back. These zombies are pure attack surface, yet confirming they are safe to delete typically takes extensive human-to-human investigation into what they do and who owns them, so the irreversible step is deferred indefinitely. The Cloud Permissions Firewall quarantines them

> **When the grant expires, the access is simply gone: nothing to revoke, nothing to recertify, nothing for an attacker to find.**

instead: an organization-level deny makes them instantly unusable to an attacker while destroying nothing. Because JIT is already the front door, revival is simple: an owner approves a request, and the identity is working again in seconds. The same process can open and close access for privileged deployment identities during change windows.

“Approved” is not unlimited: least privilege inside the session

A JIT approval opens a permission set. It does not suspend least privilege. Inside the session, privileged actions are still governed by what this user has actually done before. An SSO user approved into Administrator access who regularly changes Route53 DNS records finds those changes pre-approved the moment the session is active; but if that same user has never created an access key, the first attempt raises a secondary approval before it proceeds. Familiar workflows without friction, while a hijacked session - or an operator drifting beyond their pattern - meets a human check at the first unfamiliar privileged action.

Auto “Permissions on Demand”: On-call → SuperAdmin in one approval

That in-session least-privilege layer is a choice, made per permission set. For the scenarios where this isn’t appropriate, it can be toggled off via Auto Permissions on Demand. Interrupting a responder mid-incident with an approval for every change is how controls get bypassed, so for designated on-call and incident-response permission sets, the JIT approval itself covers all privileged activity for that session: no secondary approvals, no re-challenge when the investigation takes a new turn. One approval is the only human gate, with the session clock, scope, and record-keeping access bounded. Everywhere else, the extra checks stay on.

AI session summaries: what happened, and what looked wrong

Knowing what happened inside a time-bound session is critical for visibility and compliance. Every JIT session produces an AI-generated summary covering both what happened (ie, the services touched, the changes made) and any suspicious activity (actions outside the stated purpose of the request, or activity inconsistent with the permission set’s normal use). Approvers review a readable narrative instead of a CloudTrail export, and unusual sessions surface themselves.

Approval flows as simple - or as demanding - as the access warrants

Approval flows are configured per permission set and per scope, from self-approval — designated low-risk access granted instantly, with a full record kept — up through second-party approval. Routing can differ by time and permission set: business-hours requests to the resource owner, after-hours requests to the on-call approver, and the most sensitive permission sets always requiring a second party. Friction is calibrated to risk for the best developer experience.

What Adoption Looks Like

1. **Scope.** Decide which permission sets are sanctioned in which accounts, and quarantine the dormant identities that shouldn't be usable at all. Optionally deny everything else — enrolled in JIT or not.
2. **Delegate.** Establish owners and approvers per scope, and set the conditions: self-approval, grant time limits, second-party requirements, routing, and escalation.
3. **Enroll.** Enroll the permission sets humans should request just-in-time, including on-call permission sets covered by Auto Permissions on Demand.
4. **Operate.** Requests and approvals flow through Slack or Teams, grants expire automatically, and every session ends with an AI session summary.

Conclusion

Standing human privilege persists not because anyone defends it, but because removing it has always been a project too painful to finish. JIT in the Sonrai Cloud Permissions Firewall removes the project: no entitlement cleanup, deny-first coverage for what nobody remembered to govern, zombies quarantined instead of deleted nervously, one approval per incident, least privilege enforced even inside an approved session, and an AI-written account of every session. Access arrives in the time it takes an approver to tap a Slack message — and privilege exists only while it is in use.

Sonrai Security · sonraisecurity.com · August 2026