GUIDE

Team Onboarding and Offboarding Security Guide

Granting least-privilege access on join and revoking it completely on exit

Prepared for	Web3 teams	Classification	Public
Prepared by	Oak Security	Date	2026-06-17

Oak Security GmbH

1. What This Guide Is For

This guide is for everyone who grants or removes access in a Web3 team: founders, operations, engineering leads, and whoever currently owns the "add the new person to everything" task. It is written for mixed technical levels. You do not have to write Solidity to follow it. You have to know who gets what, when access starts, and when it ends.

The aim is not a perfect identity system. The aim is that one joiner cannot accumulate more access than their job needs, and one leaver cannot keep access after they are gone. Both failures are common, and the second is the one that drains treasuries.

> **NOTE**
>
> **Baseline rule:** grant least privilege on join, verify identity out-of-band before issuing anything, and on exit revoke every access and rotate every secret the person was exposed to within 48 hours, not "next sprint."

Offboarding is the higher-risk half. A joiner with too little access files a ticket. A leaver with lingering access is an unmonitored attacker who already knows your tooling. The Ledger Connect Kit incident was, at root, an offboarding failure: a former employee still held npm publish rights months after leaving, got phished, and the malicious package propagated to everyone who had not pinned. Treat offboarding as a security event, not an HR formality.

2. Principles That Apply to Both

Five principles sit under every checklist in this guide.

Verify explicitly. Authenticate every access grant and every removal against more than one signal. A new joiner's identity is confirmed out-of-band (a voice call to a known number, a verified reference) before any credential is issued. A "please add my new laptop" request is itself verified, because that request is exactly what an attacker sends.

Least privilege, just-in-time, just-enough. Access is granted to the task, not to the role, for the duration of the task, and no longer. No engineer automatically gets deploy rights. No operations hire automatically gets a multisig seat. No standing admin rights on production: elevated access is requested when needed, granted for a bounded window, and expires.

Separate duties. Merge and deploy are not the same person. Commit rights and deploy rights are split. Treasury and admin actions require a quorum. An insider holding both commit and deploy is a single point of compromise. That was structurally the Munchables shape, a $62 million loss through a hire who did real work for months and then drained the protocol with access they had legitimately accumulated.

Log every grant, change, and revocation. Who, when, why, witnessed. A team that cannot answer "who has access to the deployer right now?" with a definitive list does not have access control, only optimism. The audit trail is the control.

Keep the circle of trust small and named. For any sensitive action the set of people who need to know is bounded and explicit. Every extra person with access is an extra device that can be compromised and an extra path for the kill chain to cross.

3. Role-to-Access Mapping

Decide access by role before anyone joins. Default to the minimum. Escalate by request, with a named approver, for a fixed window.

Role	Standing access	Requires explicit grant (JIT, scoped)	Never
Non-technical (ops, marketing, BD)	Email/workspace, chat, docs, password manager	Specific SaaS admin as needed	Source-control org admin, production, treasury, deploy keys
Developer	Source control (scoped repos), CI read, chat, docs	Write to specific repos, staging access, short-lived deploy credentials	Long-lived deployer EOA, treasury seat, registrar, DNS, cloud root
Senior engineer / lead	The above plus CODEOWNERS on owned repos	Production access (step-up, time-boxed), upgrade-admin participation	Standing production admin, sole control of any money-moving key
DevOps / infra	Cloud (scoped IAM), CI admin, secrets-manager paths	KMS operations, DNS/registrar changes (dual-control)	Cloud root as a daily identity, treasury signing
Treasury signer	Multisig signer seat (hardware-backed), out-of-band address book	Cold-treasury participation	Deployer and admin keys on the same device, software-only signing
Founder / exec	Email/workspace, chat, docs	Break-glass access, documented and time-boxed	Convenience access to keys "just in case." The CEO does not need the upgrade-admin key

The rule behind the table: every person is on an access list because their absence is a problem, not because their presence is convenient.

4. Onboarding

The goal is a person who can do their job on day one and nothing more. Provision in order: identity, device baseline, scoped app access, then any signing role separately and later.

4.1 Before Day One

- Confirm the hire is real. Camera-on interviews, recorded. Reference calls placed by voice, not email. Live ID check on camera against the name on the paperwork. A candidate insisting on stablecoin payment to a fresh wallet and refusing standard KYC, in a role that does not need pseudonymity, is a cluster of red flags worth pausing on. The DPRK has placed engineers inside Web3 teams precisely this way.
- Write down the role-to-access grant from Section 3. The approver signs off before anything is issued.

4.2 Onboarding Checklist

1. **Create the identity.** One SSO/workspace account, unique, in the person's real name. No shared logins, ever.
2. **Enforce MFA at enrollment.** Phishing-resistant by default: passkeys or hardware security keys for critical accounts (email/workspace, source control, password manager, cloud). Not SMS. Not TOTP for production access. Verify enrollment completed before granting downstream access.
3. **Issue the password manager seat.** All work credentials live there. Set the expectation now: no secrets in Notes, chat, screenshots, shell history, or `.env` files.
4. **Confirm the device baseline.** Disk encryption on, OS updated, screen lock under five minutes, firewall on, unused sharing services off, a dedicated work browser profile separate from personal browsing and wallet activity. For critical-access roles, a dedicated hardened or single-purpose device.
5. **Grant scoped app access only.** Add to the specific repos, channels, and SaaS the role needs. Use groups so access is auditable and removable in one action. Do not add to the org-wide admin group for convenience.
6. **Issue developer credentials scoped and short-lived.** A dedicated work SSH key (hardware-backed for sensitive infra). Fine-grained personal access tokens over classic ones: a token that reads one repo cannot publish to npm. Prefer SSO plus short-lived credentials (STS, OIDC into a cloud role) over long-lived static keys.
7. **Apply least privilege from day one.** No automatic deploy rights. No automatic production access. Elevated access is a separate, time-boxed request.
8. **Record the grant.** Log every credential issued, its owner, scope, and expiry, against the named person.

> **NOTE**
>
> **Onboarding rule:** identity and MFA first, scoped app access second, any key or signing role last and separately. Never bundle a signing role into a general onboarding script.

4.3 Signer and Key Onboarding (Separate Ceremony)

Adding someone to a multisig is not a checkbox in the joining flow. It is its own ceremony, run later, deliberately.

- **One hardware wallet per signer.** Owned and held by that person alone. The private key never leaves the device. No shared device, no shared seed: two signers on one device or one seed count as one signer, and your 3-of-5 is closer to 1-of-2.
- **Separate the signer EOA from personal activity.** The signing address is not the wallet used for trading, airdrops, or random dapp connections. A signer key that has approved arbitrary contracts carries unknown allowances into your treasury.
- **Verify the new signer address out-of-band.** Confirm the full address by voice or in person against the address book, character by character or via checksum. Never by copying from chat.
- **Preserve signer diversity.** A new signer should add diversity, not reduce it, across four axes: geography, department, device vendor, and signing UI path. A 4-of-7 where everyone uses the same UI on the same device class in the same country is a 1-of-1 against a single-axis compromise.
- **Adding an owner is a privileged on-chain change.** Propose, simulate, verify the calldata on hardware, sign to threshold, confirm on a block explorer. Treat owner changes as the most security-sensitive transactions a Safe processes.

5. Periodic Access Reviews

Onboarding grants drift. People change roles, projects end, "temporary" access becomes permanent. Review on a cadence, not only at exit.

- **Quarterly access review.** For every person, confirm their access still matches their current role. Remove anything no longer used. Expire stale tokens, unused SSH keys, and dormant SaaS admin grants.
- **Review on every role change.** A developer becoming a signer, or a signer leaving the multisig, triggers a re-grant and a rotation of every credential the change affects. New access is added; old access is removed in the same action, not left to accumulate.
- **Reconcile keys against the estate.** Every key has a named owner, a storage tier, a rotation cadence, and a blast-radius tier. A key with none of these is not under custody. The keys teams forget are infrastructure and developer credentials (cloud and KMS, DNS, GitHub PATs and SSH keys, npm/PyPI/Cargo publish tokens, CI service accounts), not the treasury multisig everyone remembers.
- **The insider angle.** Periodic review is the control that catches privilege accumulation, the precondition for the insider attack. Combine it with separation of commit and deploy rights and a two-person rule on production. Keep a red-flag log: unusual asks, camera-off patterns, claims that did not check out. Some insider attacks signal for weeks, and nobody remembers unless somebody wrote it down.

6. Offboarding

Offboarding must be fast and complete. Partial offboarding is the dangerous state: a leaver with one lingering credential is an attacker who already knows your systems. Run the full sequence within 48 hours of departure, faster for an involuntary or hostile exit. For a hostile exit, revoke before notifying.

6.1 Onboarding vs Offboarding at a Glance

Action area	Onboarding	Offboarding
SSO / identity	Create account, enroll MFA	Suspend account, kill active sessions, revoke MFA devices
App access	Grant scoped group membership	Remove from every group; verify no orphaned direct grants
Developer credentials	Issue scoped SSH key and fine-grained tokens	Revoke SSH keys, PATs, npm/registry publish tokens, CI accounts
Shared secrets	N/A (avoid shared secrets)	Rotate every shared secret and API key the person knew
Signing role	Separate ceremony, later	Remove from multisig, rotate threshold if needed
Devices	Confirm baseline	Reclaim hardware, wipe or rebuild, confirm no local secrets remain
Audit	Log the grant	Log every revocation; confirm against the onboarding record

6.2 Offboarding Checklist

9. **Revoke SSO and app access first.** Suspend the workspace/SSO account. This cuts the widest path. Then remove the person from every group across source control, chat, docs, hosting, cloud, registrar, and every SaaS. Verify there are no direct (non-group) grants left behind.
10. **Audit and kill lingering sessions.** Suspending the account is not enough: existing OAuth grants, app passwords, API sessions, and refresh tokens can outlive it. Force sign-out everywhere, revoke OAuth app authorizations, and review connected third-party apps. Long-lived sessions are how a "removed" user stays in.
11. **Revoke developer and deploy keys.** Remove the person's SSH keys from source-control accounts and servers. Revoke their personal access tokens, npm and package-registry publish tokens, CI service-account access, and container-registry pushes. This is the Ledger Connect Kit lesson: publish rights outliving employment is how a single phished leaver poisons a supply chain.
12. **Rotate every shared secret the person was exposed to.** Not just their personal credentials. Any API key, RPC provider key, database credential, service token, Wi-Fi/VPN secret, or password they could have read must be rotated, because they may have copied it. If you do not know what they could see, assume the worst and rotate broadly. This is why shared secrets are an anti-pattern: each one multiplies offboarding work.
13. **Remove from every multisig and adjust the threshold if needed.** Covered in detail in Section 6.3.
14. **Reclaim and rebuild hardware.** Recover laptops, hardware wallets, hardware security keys, and any company-owned device. Wipe or rebuild before reissue. Confirm no company secrets remain on personal devices, including anything synced into a personal iCloud/cloud account during employment.
15. **Review what synced where.** If work credentials touched a personal password manager, iCloud Keychain, or personal cloud storage, incident cleanup extends to those, not only company-owned systems.
16. **Log and confirm.** Check the offboarding actions against the onboarding record, credential by credential. The onboarding log is your checklist for what to revoke. If you did not log issuance, you cannot prove complete revocation.

> **NOTE**
>
> **Offboarding rule:** suspend SSO, kill lingering sessions, revoke deploy and publish keys, rotate every shared secret the leaver knew, remove from multisigs and re-threshold, reclaim devices. All of them, within 48 hours.

6.3 Removing a Signer from a Multisig

A departing signer is a key you no longer control. Treat the seat as compromised the moment the person leaves, even on a friendly exit.

- **First, confirm the honest signers still comfortably hold the threshold.** The owner-removal transaction itself consumes a threshold of signatures, and the departing key must not be the one needed to reach it.
- **Swap or remove the owner.** Propose an owner removal or `swapOwner` to a fresh, verified address on a clean device. Every remaining signer verifies the calldata on their own hardware, confirms the new owner set out-of-band, and signs. Confirm the change on a block explorer.
- **Adjust the threshold if the signer count dropped.** Removing a signer from a 3-of-5 leaves a 3-of-4, which is fine for safety but thinner on availability. Removing from a 2-of-3 leaves a 2-of-2, which fails to availability the moment one more signer is unreachable. Re-threshold deliberately as part of the same change; do not silently leave a degraded configuration.
- **Hostile exit: evacuate before you rotate.** If you cannot trust the departing signer and they could plausibly reach the threshold with one more key, do not race them with a rotation. Move funds to a clean, freshly verified Safe with a fresh signer set first. Rotation is a race the attacker can front-run. A clean Safe is recoverable; a drained one is not.
- **Rotate anything the signer touched beyond the seat.** If they had a deployer key, an admin role, or shared infrastructure access, rotate those on the same timeline. Removing the multisig seat does not address the rest of their access.

7. Common Mistakes

These are the recurring failures behind most onboarding and offboarding incidents.

- **Bundling a signing role into onboarding.** A multisig seat is a ceremony with out-of-band address verification, not a line in a joining script.
- **Granting role-wide access for convenience.** Adding the new hire to the org admin group because it is faster than scoping. This is how privilege accumulates into an insider risk.
- **Standing admin rights.** Permanent production admin instead of just-in-time, time-boxed elevation.
- **Shared logins and shared secrets.** A shared account cannot be attributed, and a shared secret multiplies the rotation burden on every exit.
- **Slow offboarding.** "We will clean it up next sprint." A leaver with lingering access is an attacker who knows your systems. The window is 48 hours, not weeks.
- **Suspending the account but leaving sessions alive.** Active OAuth grants, app passwords, and refresh tokens outlive a disabled account. Kill them explicitly.
- **Forgetting deploy and publish keys.** SSH keys on servers, npm publish tokens, CI service accounts. The Ledger Connect Kit failure mode.
- **Not rotating shared secrets a leaver knew.** Removing their account does nothing to a database password or API key they already copied.
- **Removing a signer without re-thresholding.** Silently leaving a 2-of-2 or a degraded availability margin.
- **Racing a rotation against a possibly-compromised signer.** On a hostile exit, evacuate funds first; do not give the attacker a transaction to front-run.
- **No log of what was issued.** Without an onboarding record you cannot prove complete revocation. The audit trail is the control.

8. References

- NIST SP 800-207, Zero Trust Architecture: https://csrc.nist.gov/pubs/sp/800/207/final
- NIST SP 800-57 Part 1 Rev 5 (key management, cryptoperiods): https://csrc.nist.gov/pubs/sp/800/57/pt1/r5/final
- CISA Zero Trust Maturity Model v2.0: https://www.cisa.gov/zero-trust-maturity-model
- Safe documentation (owners, threshold, swapOwner): https://docs.safe.global/
- GitHub fine-grained personal access tokens: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens
- Ledger Connect Kit incident write-up: https://www.ledger.com/blog/a-letter-from-ledger-chairman-ceo-pascal-gauthier-regarding-ledger-connect-kit-exploit