SETUP GUIDE

Multisig Treasury Operations Setup Guide for Web3 Teams

Designing and operating a Safe multisig without blind-signing

Prepared for	Web3 team members	Classification	Internal
Prepared by	Oak Security	Date	2026-06-17

Oak Security GmbH

1. What This Guide Is For

This guide is for everyone involved in custody of organizational funds: treasury signers, finance, operations, and the engineers who set up and maintain the multisig. Some of you sign transactions but do not write contracts. That is fine. The signing discipline in this guide does not require you to read Solidity. It requires you to compare what a screen tells you against what you expected, and to refuse when they disagree.

The aim is not a perfect setup. The aim is that no single mistake can move organizational funds on its own, whether that mistake is a phished signer, a swapped clipboard address, a malicious proposal, or a compromised front end.

> **NOTE**
>
> **Baseline rule:** use a threshold above one with genuinely independent signers, one hardware wallet per signer, have every signer independently verify the calldata, and never blind-sign.

Most large multisig losses were not breaks of the underlying contract math. They were signers approving something they did not understand, because they trusted an interface instead of verifying on their own hardware. Treat the signing interface as untrusted. Treat the hardware screen as the source of truth.

2. Multisig Design

Safe (formerly Gnosis Safe) is the reference implementation and the assumed platform throughout this guide. The principles transfer to other multisigs, but the concrete steps below are written for Safe.

Signer Count and Threshold

The threshold is the number of signatures required to execute. The signer set is the total number of keys that can contribute a signature. Write it as threshold-of-total, for example 3-of-5 or 4-of-7.

Two failures pull in opposite directions. A threshold that is too low fails to safety: one or two compromised keys move funds. A threshold that is too high fails to availability: a lost device, a traveling signer, or an unreachable colleague can freeze the treasury. Design for both.

Configuration	Protects Against	Weakness
2-of-3	Single-key compromise. Workable for a small team or a hot operational Safe.	Two correlated failures (two signers in one office, two keys on one backup) defeat it. Thin availability margin.
3-of-5	Single-key compromise plus one unavailable signer. A reasonable default for most treasuries.	Requires five genuinely independent signers.
4-of-7	Up to three compromised keys, and up to three unavailable signers. Suited to large or cold treasuries.	Coordination overhead. Needs seven people who can each verify and sign competently.

A threshold protects against single-key compromise only to the extent that the keys are independent. Five keys held by two people, or backed up to one cloud account, is not 3-of-5. It is closer to 1-of-2.

Signer Independence

Independence is the property that makes the threshold real. Maximize it across three axes:

- **People.** Different individuals, not one person holding several keys.
- **Devices.** A separate hardware wallet per signer. No shared devices, no shared seed.
- **Location.** Signers in different offices, cities, or jurisdictions so a single physical event (theft, raid, fire, coercion) cannot reach the threshold.

The attack class to design against is correlated compromise: one phishing campaign, one malicious browser extension, one supply-chain incident, or one physical event that takes several signers at once. Independence breaks the correlation.

Hot and Cold Separation

Do not run day-to-day operations and long-term reserves from the same Safe. Split them.

- **Hot operational Safe.** Lower balance, lower threshold (for example 2-of-3 or 3-of-5), used for payroll, vendor payments, and routine operations. Funded periodically from cold storage.
- **Cold treasury Safe.** The bulk of funds, a higher threshold, a wider and more geographically spread signer set, used rarely and only with deliberate process. Optionally behind a timelock.

This limits blast radius. A compromise of the operational Safe loses an operating buffer, not the treasury.

Modules and Guards

Safe is extensible through modules and guards. Understand the difference, because the security implications are opposite.

- **Modules** can execute transactions on the Safe bypassing the normal owner-threshold check. A module is, by design, a privileged path around your multisig. An over-privileged or unaudited module is a single point of failure that defeats the entire threshold.

- **Guards** sit in the transaction execution path and can block or constrain transactions that would otherwise pass. A guard can enforce policy (allow-lists, value limits), but a buggy or malicious guard can also brick the Safe by reverting every transaction.

Both are covered in Section 6. The short version: add modules and guards sparingly, only when audited, and treat every one as part of your trusted computing base.

3. Quick Setup Checklist

These are the decisions and settings every treasury Safe should have before it holds meaningful funds.

Deployment and Configuration

Item	What To Do	Where To Check
Network and address	Deploy on the intended chain. Record the Safe address. Verify it independently before funding.	Safe app, block explorer
Signer set	Add only the agreed signer addresses. Verify each address out-of-band against the address book.	Safe settings (owners), confirmed on chain
Threshold	Set the agreed threshold (for example 3-of-5). Confirm it on chain, not just in the UI.	Safe settings (threshold), confirmed on chain
Implementation	Confirm the Safe uses the expected canonical singleton and proxy factory for the network.	Block explorer, Safe deployment records

Signers and Policies

Item	What To Do
Hardware signers	Every signer uses a dedicated hardware wallet. No software-only keys for treasury signing.
Address book	Maintain a verified, out-of-band address book of all signer addresses and frequent payees.
Spending limits	If using the hot Safe for routine payments, configure allowance limits rather than full-threshold signing for small amounts.
Guards and modules	Install only audited guards/modules. Document

	every one, its purpose, and its privilege.
Recovery	Decide and document the recovery path (timelocked recovery, social recovery, or a documented signer-rotation process) before you need it.

Monitoring and Process

Item	What To Do
Monitoring	Enable alerting on every transaction, owner change, threshold change, and module change on the Safe.
Simulation	Require transaction simulation (for example via Tenderly) before signing anything non-trivial.
Runbook	Write down the propose-simulate-verify-sign-execute workflow and the incident steps. Every signer reads it.
Test transaction	Run a small end-to-end test before moving real value through a new Safe or a changed configuration.

4. Signer Setup and Key Diversity

Each signer is an independent control. The setup has to preserve that independence.

Per-Signer Requirements

- **One hardware wallet per signer.** A Ledger, Trezor, or equivalent, owned and held by that signer alone. The private key never leaves the device.
- **No shared devices and no shared seed.** Two signers must not be backed by the same physical device or the same recovery phrase. If they are, the threshold counts them as one.
- **Separate signer EOAs from personal activity.** The address a signer uses for the Safe should not be the wallet they use for trading, airdrops, minting, or random dapp connections. A signer key that has approved arbitrary contracts is a signer key with unknown allowances and a wide attack surface.
- **Secure the seed backup.** Each signer's recovery phrase is stored offline, never photographed, never typed into a computer, never synced to cloud. Backups for different signers are stored in different places so one location does not yield the threshold.

The Out-of-Band Address Book

Maintain an address book of every signer address (and frequent payee) that was verified through a channel other than the one used to propose transactions. Verify addresses by voice or in person, character by character or via a checksummed comparison, never by copying from a chat message or a web page.

The attack this defeats is address substitution: a malicious front end, a compromised chat, or a clipboard hijacker swapping a signer or payee address for the attacker's. If the only place an address lives is the interface you are signing in, you cannot detect the swap.

Avoiding Correlated Compromise

- Buy hardware wallets from the manufacturer or an authorized reseller, not a third-party marketplace.

- Do not have all signers configure their devices on the same machine or the same network at the same time.
- Do not install the same risky browser extensions across signer machines.
- Keep signing-device firmware updated, but verify update prompts on the device, not from a pop-up in a browser.

5. The Signing Workflow

The workflow is the same for every transaction, large or small: **propose, simulate, independently verify, sign, execute.**

The discipline that matters is in the verify step, and it happens on each signer's own hardware screen, not in the proposing interface. The interface can lie. The hardware screen shows what you are actually authorizing.

The Five Steps

1. **Propose.** One person constructs the transaction in the Safe interface and shares it with the signers.
2. **Simulate.** Run the transaction through a simulator (for example Tenderly, or Safe's built-in simulation) to see its effects: balance changes, approvals granted, state changes. A simulation that shows an unexpected transfer or approval is a stop signal.
3. **Independently verify.** Each signer checks the transaction details, including on their hardware device. See the checklist below. Verification is independent: signers do not just trust that the proposer or a previous signer already checked.
4. **Sign.** Only after verification passes. If anything does not match, do not sign and raise it out-of-band.
5. **Execute.** Once the threshold is reached, submit. Confirm execution on a block explorer.

What Each Signer Must Verify on the Hardware Screen

Trust the device, not the dApp UI. On the hardware wallet, confirm:

Field	Check
Safe address	The transaction is for your Safe, on the expected chain. A correct-looking UI can be pointed at a different Safe.
To / target	The target contract or recipient matches the address book. Verify the full address, not the first and last four characters.

Value	The native-token amount is what you intended (often zero for contract calls).
Function selector	The 4-byte selector and decoded function match the intended action (for example `transfer`, `approve`, `execTransaction`).
Decoded calldata	Parameters match: recipient, amount, token, spender, and any nested call data.
Operation	`Call` versus `delegatecall`. See below.
Nonce	The nonce matches the specific proposal you intend to approve. Two different transactions can be queued at the same nonce, and only one will execute, so an attacker can race a malicious transaction against the one you expect. A signature for a future nonce also stays valid and executes whenever the Safe reaches that nonce. Confirm no competing proposal shares the nonce.

Critical caveat: you are not signing an ordinary transaction. You are signing an EIP-712 `SafeTx` typed-data struct, and most hardware wallets do not recursively decode the inner `data` field. The device typically shows the `SafeTx` hash plus the top-level `to`, `value`, `data`, and `operation`, but not the decoded token transfer or the contents of a MultiSend batch. If the device cannot show you the decoded inner call, do not assume the proposing interface is telling the truth about it. Independently compute the `SafeTx` hash (domain separator plus message hash) with a tool that does not depend on the proposing interface, for example a local `safe-tx-hashes` utility, and confirm it matches the hash on the device before you approve. This mismatch between what the interface claimed and what was actually signed is exactly the Bybit failure mode.

Disable "blind signing" on the signing device (the Ledger setting is literally named "Blind signing", and enabling it permits signing opaque data, which is the opposite of what you want) and enable contract-data / clear-signing so the device decodes EIP-712 fields where it can. If the device can only show an opaque hash and you have not independently verified the `SafeTx` hash, do not sign.

The delegatecall Danger

Safe supports two operation types: a normal `Call` and a `delegatecall`. A `delegatecall` runs the target contract's code in the Safe's own context, with the Safe's storage and permissions. A malicious or buggy `delegatecall` target can rewrite the Safe's owners, threshold, or modules, or drain it outright.

Routine treasury transactions (token transfers, approvals, standard contract interactions) are `Call`. A `delegatecall` in a proposal you did not expect is a red flag. Anyone proposing one must explain exactly why, and every signer must understand the target before signing. Batching tools (MultiSend) legitimately use `delegatecall`, so verify the MultiSend contract is the canonical one and inspect every inner transaction.

Message Signing (EIP-712 and EIP-1271)

Signers also approve off-chain messages, for example to log into a dApp, list an NFT, or authorize an order. These use EIP-712 typed data. A Safe, being a contract, proves message validity through EIP-1271 (`isValidSignature`).

Off-chain signatures are a known attack vector. A malicious site can ask you to sign a typed-data message that is actually an open token approval or an order that transfers assets. Apply the same discipline: read the typed-data domain and fields, confirm what the message authorizes, and refuse if it does not match what you intended to do. A signature that grants a `setApprovalForAll` or an unlimited `approve` is as dangerous as an on-chain transfer.

6. Policies, Modules, and Guards

These mechanisms let you encode policy on chain. They also expand the trusted computing base. Every addition is something that can fail or be abused.

Spending and Allowance Limits

For a hot operational Safe, the Safe allowance module lets you grant a signer (or an automation) the right to spend up to a capped amount of a specific token per time period without collecting the full threshold each time. This reduces friction for payroll and small vendor payments. Scope it tightly: specific token, specific recipient where possible, conservative limit, short period. The allowance holder becomes a lower-threshold path, so size the cap to what you can afford to lose.

Transaction Guards

A guard runs on every execution and can revert transactions that violate policy: enforce a recipient allow-list, cap per-transaction value, or block certain selectors. A guard is a constraint, not a privilege escalation, but it sits in the critical path. A buggy guard that reverts unconditionally bricks the Safe until removed, and removing it itself requires a passing transaction. Test guards on a throwaway Safe first, and keep the removal path verified.

Recovery Modules and Timelocks

- **Recovery modules** (for example social recovery or a delay-based recovery) provide a path back if signers are lost. They are also a privileged path in: whoever controls recovery can eventually control the Safe. Set delays long enough to detect and contest a malicious recovery.
- **Timelocks** delay execution of sensitive actions (upgrades, owner changes, large transfers), giving signers and monitoring time to react to an unexpected queued transaction. For a cold treasury, a timelock on owner and threshold changes turns a silent takeover into a visible, contestable event.

Role Separation

Where the tooling supports it, separate who can propose, who can sign, and who can execute. Proposal rights are low-privilege (a proposer cannot move funds alone). Reserve signing rights for the hardware-backed signer set.

The Risk of Modules

Repeat, because it is the most common way a well-configured Safe is lost: a module can move funds bypassing the threshold entirely. The questions for every module:

- Is it audited, and by whom? Can you read the report?
- What exactly can it do to the Safe, and under whose authority?
- Who can call it, and can that caller be compromised independently of the Safe?
- What is the removal path if it misbehaves?

If you cannot answer all four, do not install it. An over-privileged or unaudited module is a single point of failure that makes your threshold cosmetic.

7. The Recommended Setup Flow

Step 1: Design the Threshold and Signer Set

Agree the threshold and the signer set against Section 2. Pick people who are genuinely independent and who can each verify a transaction competently. Decide hot versus cold split. Write the design down before touching any tooling.

Step 2: Each Signer Completes Hardware Setup

Every signer sets up their own hardware wallet (Section 4): genuine device, offline seed backup, a signer EOA kept separate from personal activity, clear-signing enabled. Signers do not set up together on one machine.

Step 3: Deploy the Safe and Verify It

Deploy the Safe on the intended network. Then verify, independently of the deploying interface:

- The Safe address, on a block explorer.
- That it uses the canonical Safe singleton and proxy factory for that network.
- The initial owners and threshold as recorded on chain.

Do not fund a Safe you have not verified on chain.

Step 4: Add Signers and Verify Each Address Out-of-Band

Add the agreed owner addresses. For each one, confirm the full address through a separate channel (voice or in person) against the out-of-band address book. Confirm the final owner set and threshold on chain match the design from Step 1.

Step 5: Set Policies and Guards

Configure allowance limits for the hot Safe if used. Install only audited guards and modules, documenting each one’s purpose, privilege, and removal path. Test any guard or module on a throwaway Safe before applying it to the real one.

Step 6: Run a Small End-to-End Test Transaction

Before moving real value, run a small transaction through the full workflow: propose, simulate, every signer verifies on hardware, sign to threshold, execute, confirm on explorer. This validates the signer set, the threshold, the devices, and the monitoring at once.

Step 7: Document the Runbook and Enable Monitoring

Write the runbook: the signing workflow, who proposes, how proposals are shared, the verification checklist, and the incident steps from Section 10. Enable monitoring and alerting on every transaction, owner change, threshold change, and module change. Confirm at least one person receives and reads the alerts.

8. Extra Rules for Critical-Access Users

For large treasuries, where a single bad signature is a company-ending event, the baseline is not enough. Apply the stricter controls below.

Control	Requirement
Geographic spread	Distribute signers across cities, ideally jurisdictions, so no single physical or legal event reaches the threshold.
Jurisdictional spread	For very large treasuries, ensure no single legal authority can compel enough signers to meet the threshold.
Air-gapped signing	Use air-gapped hardware wallets (QR or SD-card transfer) for the cold treasury, so signing devices never touch an internet-connected machine. Air-gapped flows are the weakest at decoding nested Safe calldata, so the independent `SafeTx` hash verification from Section 5 is mandatory here, not optional.
Mandatory simulation	Every cold-treasury transaction is simulated and the simulation results are reviewed by signers before signing. No exceptions for "obvious" transactions.
Change management	Treat any change to the signer set, threshold, modules, or guards as a privileged change with its own proposal, review, and out-of-band verification. Owner changes are the most security-sensitive transactions a Safe processes.
Dual control for upgrades	Require more than one reviewer to sign off, out-of-band, before a signer change, threshold change, or module/guard change is proposed.
Timelock on sensitive actions	Put owner changes, threshold changes, module changes, and large transfers behind a timelock so an unexpected one can be detected and contested before it executes.

Separate signing material	Do not store the backups of multiple cold-treasury signers in the same location. Do not travel with material that, combined, reaches the threshold.
Independent verification	At least two signers must decode and verify calldata independently for cold-treasury transactions. Do not let one engineer's "it's fine" stand in for the group.

9. Things To Avoid

These are the patterns behind most multisig losses.

- **Blind-signing.** Approving a transaction or message shown only as a hash, or trusting a UI summary instead of the decoded calldata on the hardware screen.
- **Trusting the dApp UI over the hardware screen.** The front end is the least trustworthy element in the chain. The hardware screen is the source of truth.
- **One device or one seed backing multiple signers.** This collapses the threshold. 3-of-5 across two devices is not 3-of-5.
- **A threshold that is too low.** 1-of-n is not a multisig. 2-of-n with two signers in one office is barely better.
- **All signers in one office or one jurisdiction.** Correlated physical and legal risk.
- **Unverified or over-privileged modules.** A module that bypasses the threshold is a backdoor by design. Unaudited modules and guards belong nowhere near a treasury Safe.
- **Reusing signer keys for personal activity.** A signer EOA that connects to random dapps carries unknown approvals and a wide attack surface into your treasury.
- **Approving an unexpected `delegatecall`** without understanding the target.
- **Signing off-chain EIP-712 messages without reading them.** An open approval signed off-chain drains the same as an on-chain transfer.
- **No out-of-band address verification.** If the only source for a payee or signer address is the interface you are signing in, you cannot catch a substitution.
- **No monitoring on owner/threshold/module changes.** A silent takeover you do not see is one you cannot contest.

10. If Something Suspicious Happens

The trigger is a proposal you did not expect, a proposal whose decoded calldata does not match its description, a suspected signer compromise, or a signing request from an interface that looks wrong. The first rule is simple: **do not sign.**

Immediate Actions

6. **Do not sign.** A transaction that is not signed to threshold cannot execute. Withholding your signature is the control.
7. **Alert the other signers out-of-band.** Use a channel other than the one the proposal came through. Tell them not to sign.
8. **Investigate the proposal's origin.** Who proposed it? Through what interface? Does the decoded calldata match the stated intent? Does a simulation show unexpected transfers, approvals, owner changes, or a `delegatecall`?
9. **Check for owner, threshold, and module changes.** A malicious proposal often tries to add an owner, lower the threshold, or install a module rather than transfer funds directly.

If a Signer May Be Compromised

- Have the affected signer stop signing immediately.
- Assess whether the compromised key (or keys) could reach the threshold. If the attacker now needs only one more key, or already controls enough to sign alone, treat funds as actively at risk. Evacuate them to a clean, freshly verified Safe with a fresh signer set FIRST, before attempting rotation. Rotation is a race the attacker can win or front-run.
- Only if the remaining honest signers comfortably hold the threshold, rotate the affected signer: propose and execute an owner swap (`swapOwner`) to a fresh, verified address on a clean device, verified out-of-band by the remaining signers. Note that the owner swap itself consumes a threshold of signatures, and the compromised key must not be the one needed to reach it.
- When in doubt, prefer migrating funds to a new Safe over leaving them under a possibly-compromised owner set. A clean Safe is recoverable; a drained one is not.

The General Lesson

The largest multisig losses to date did not come from breaking the Safe contract. They came from manipulating what signers saw at signing time: a tampered front end or compromised signing infrastructure presenting one thing on screen while collecting signatures for another. The control that would have caught it is the same one in Section 5: independent verification of the decoded transaction on each signer's own hardware screen. On-chain verification on the device is the control that matters. Everything upstream of the hardware screen can be compromised.

11. Monthly Self-Check

Once a month, run this review across the signer group.

Check	Done
The threshold and signer set still match the documented design.	
Every signer still controls their own hardware wallet, and no device or seed backs two signers.	
The out-of-band address book is current and was used to verify recent payees.	
No unexpected owners, threshold changes, modules, or guards appear on the Safe.	
Every installed module and guard is still audited, documented, and justified.	
Allowance limits on the hot Safe are still correctly scoped.	
Signers have confirmed they verify calldata on their hardware screen and do not blind-sign.	
"Blind signing" is disabled and contract-data / clear-signing is enabled on every signing device.	
Transaction simulation is being run before non-trivial transactions.	
Monitoring and alerting on transactions and configuration changes is working and someone reads the alerts.	
The recovery path and incident runbook are current and known to every signer.	
Hot and cold Safes still hold appropriate balances (operating buffer vs reserves).	

12. References

- Safe documentation: https://docs.safe.global/
- Independent SafeTx hash verification utility (safe-tx-hashes): https://github.com/pcaversaccio/safe-tx-hashes-util
- Safe Help Center (verifying transactions on a hardware wallet): https://help.safe.global/
- Safe modules and guards overview: https://docs.safe.global/advanced/smart-account-modules
- Safe allowance module (source): https://github.com/safe-global/safe-modules/tree/main/modules/allowances
- EIP-712 (Typed structured data signing): https://eips.ethereum.org/EIPS/eip-712
- EIP-1271 (Standard signature validation for contracts): https://eips.ethereum.org/EIPS/eip-1271
- Tenderly transaction simulation: https://docs.tenderly.co/simulations
- Safe post-mortem on the Bybit signing-interface attack: https://safe.global/blog/recovering-and-rebuilding-after-the-bybit-attack