GUIDE

Zero Trust Architecture: An Introduction for Web3 Teams

Why the perimeter died, what zero trust actually is, and where to start

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 Web3 founders, operators, and engineers who keep hearing "zero trust" and want to know what it means, what it does not mean, and how it applies to a distributed team that holds keys and runs a treasury.

It is conceptual, but it stays concrete. By the end you should understand the architecture well enough to read every other control in your opsec program as an instance of it: phishing-resistant MFA, multisig design, device hardening, deployment authority. Those are not separate topics. They are the same posture applied to different assets.

The core claim is simple. The perimeter model that most security tooling still assumes does not fit a Web3 startup, and it never will. Zero trust does. The work is to understand why, and then to apply it incrementally instead of waiting for a security team you do not have.

> **NOTE**
>
> **Mental model:** never trust, always verify. No user, device, workload, or network is trusted by default, regardless of where it sits. Trust is granted per request, against current context, and it expires.

2. Why the Perimeter Model Died

The traditional model is castle-and-moat. A hard outer wall (corporate firewall, VPN gateway, the office building) surrounds a soft, trusted interior. Inside the wall, users and devices are trusted. Outside, everything is hostile. The security job is to make the wall thick.

That model worked while its assumptions held: everyone in the office, a known set of subnets, corporate-owned and corporate-imaged endpoints. The perimeter was real, small, and drawable on a whiteboard.

Five shifts dissolved it. The cloud moved workloads off the corporate network. SaaS moved primary work tools to applications the network does not control. Remote work moved users outside the building. BYOD moved endpoints onto devices the organization does not own. Contractors and third-party integrations brought external identities directly into the trusted interior.

Once the wall dissolves, the failure mode is lateral movement. An attacker who breaches whatever wall remains (a VPN gateway, a phished laptop) finds that everything inside still trusts everything else by default, and looks indistinguishable from an insider because the model never had a way to tell the difference. Target (2013), the OPM breach (2015), and SolarWinds (2020) all followed this pattern: the initial compromise was bad, the lateral movement that followed was catastrophic, and it was possible only because the interior was implicitly trusted.

The lesson across two decades of incidents: the perimeter has dissolved, and identity and granular access are the new perimeter.

2.1. The Web3 Startup Reality

A Web3 organization looks nothing like the enterprise the perimeter model was built for. The team is distributed across time zones with no central office. The hierarchy is flat. The pace is fast. Contractors are normal, not exceptional. Devices are heterogeneous: a personal Mac, a Linux box at home, a cloud dev box, all touching the same codebase. Everything from source control to chat to identity sits in SaaS.

Security policy is thin. There is rarely a CISO, rarely a written acceptable-use policy. Managed laptops, mandatory VPN, locked-down browsers: the operational cost is more than a team shipping by Friday can absorb, even when they agree the controls are necessary. Run down the standard enterprise checklist (managed devices, firewall, corporate PKI, uniform policy enforcement, endpoint monitoring, a CISO) and the honest answer for the median Web3 team is "no" to almost all of it. The one exception is MFA, and even there the form matters.

This is the environment zero trust was designed for. The description (distributed team, heterogeneous devices, network is the public internet, no perimeter) fits a Fortune 500 cloud migration. It also fits a fifteen-person Web3 team operating from six countries, almost without modification.

3. What Zero Trust Is (and Is Not)

Zero trust is an architecture, not a product. The term was coined by John Kindervag at Forrester in 2010 and formalized in 2020 in NIST SP 800-207, *Zero Trust Architecture*, the closest thing the field has to a canonical, vendor-neutral reference.

The core tenet: no user, device, workload, or network is trusted by default, regardless of location, on or off any corporate network. Location-independence is the radical part. A request from inside the office is treated identically to one from a coffee shop. A request from a corporate laptop is treated identically to one from a contractor's personal device, except where the contextual signals differ. There is no privileged "inside."

Every access request is authenticated, authorized, and continuously validated. Authentication establishes who or what is making the request. Authorization establishes whether that identity may perform the requested action on the requested resource under current context. Continuous validation means the decision is re-evaluated against current signals as the session proceeds, not made once and cached.

When trust is granted, it is granted per request, contextually, and it expires. Sessions are short, tokens rotate, and elevated privileges in particular are time-bounded.

The misunderstandings matter, because acting on them produces worse security than acting on no plan at all:

- **Not just MFA.** MFA is one enforcement control. Requiring phishing-resistant MFA on logins while letting any authenticated user do anything inside the application implements one signal of "verify explicitly" and none of least privilege or assume breach.
- **Not a VPN replacement.** Zero Trust Network Access (ZTNA) is one slice of the architecture, a useful one. Swapping a VPN for a ZTNA product and leaving the rest untouched is a network-layer improvement and an incomplete implementation.
- **Not a product you buy.** Almost every security product is now marketed as "zero trust." The claim is often nominally true (the product implements some piece) and misleading insofar as it implies the purchase completes the architecture. Zero trust spans identity, devices, networks, applications, workloads, data, and the policies connecting them.

- **Not "trust nobody, ever."** Trust is granted under zero trust. What is rejected is implicit, persistent trust: the assumption that because something was trusted once, or sits on a particular network, it should be trusted again now without further evaluation.
- **Not a one-time project.** It is a continuous posture and a multi-year journey. The CISA Zero Trust Maturity Model describes stages (traditional, initial, advanced, optimal) across pillars, and most organizations sit at different maturity levels in different pillars at once.

4. The Three Core Principles

NIST and the major implementations distil zero trust into three principles. They are partial individually and stronger as a set: verify-explicitly gives least-privilege something to evaluate, least-privilege limits the consequences when verify-explicitly is wrong, and assume-breach catches the cases where the first two fail.

Principle	What it means	Web3 example
Verify explicitly	Authenticate and authorize every request using every signal available: identity, device posture, location, workload identity, data sensitivity, behavioral anomalies. No single signal, including identity, is sufficient alone.	A signer authenticated correctly but on an unmanaged device at 3am, approving an ownership transfer, is not the same trust scenario as that signer on their normal device during working hours.
Use least-privileged access	Access is just-in-time and just-enough: granted for the specific task, for its duration, and no longer. The direct consequence is no standing admin rights.	An engineer does not hold permanent admin on production. They request elevated deploy access, it is evaluated against current signals, granted for a bounded window, and it expires.
Assume breach	Plan as though one or more components are already compromised. Minimize blast radius by design. Segment access, encrypt end-to-end, use analytics to detect early.	A single compromised laptop must not cascade into a drained treasury. One phished signer must not, by their action alone, move funds.

4.1. The Assume-Breach Mindset

Assume-breach is the principle most likely to get lip service and least likely to be taken seriously. It is also the one that most cleanly separates a real posture from a nominally compliant one.

> **NOTE**
>
> **Mental model:** assume you have already been breached, then design so that this assumption does not produce catastrophe. Some device is compromised. Some session token is stolen. Every inbound contact (an email from a known correspondent, a Telegram message from a partner, a Discord ping from a contributor) is malicious until proven otherwise.

The goal is not to prevent breach. It is to make breaches harmless. A single compromised device must not cascade into a compromised treasury. A single phished signer must not drain the multisig. The keys must not sit where one laptop's malware can steal them. One signer's bad day must not be the protocol's worst day.

The Bybit incident is the canonical recent illustration. The proximate compromise was a Safe{Wallet} frontend injection that lied to signers about the transaction they were approving. Several signers, on several devices, all approved it. The breach was, in some sense, irreducible: the signers did what they were supposed to, with the tools they were supposed to use, and the tools deceived them. The architectural question is therefore not "how do we prevent the signers from being deceived" (a question with no fully reliable answer) but "how do we contain the consequences when they are." That is what defense-in-depth means: a second control behind the deceived signers (a transaction simulator on a separate device, a hardware-wallet display of the actual call data, an out-of-band approval workflow, a delay window).

5. The NIST Reference Architecture: PDP and PEP

Zero trust has a small but important vocabulary, drawn primarily from NIST SP 800-207. The two terms you will meet most often are the Policy Decision Point (PDP) and the Policy Enforcement Point (PEP).

The PDP decides, for a given access request, whether it is permitted under current policy and current context. It takes inputs from many sources (the identity provider for who the principal is, the device-posture service for compliance state, a threat-intelligence feed for anomaly indicators, the resource catalogue for target sensitivity) and produces a decision: allow, deny, allow with conditions, or step up.

The PEP enforces the decision. It sits in the path of the request and either lets it through, blocks it, or imposes the conditions the PDP returned (requiring an additional factor, restricting the scope of access). PEPs are typically distributed, one in front of each protected resource. PDPs are typically more centralized, often a single policy engine for the organization.

Separating decision from enforcement is the architecturally important part. The PDP can be improved (better signals, better policy, better threat data) without touching every PEP. PEPs can be added to new resources as they come online without changing the central policy. That separation is what makes zero trust incrementally deployable, which matters for an under-resourced team that cannot rebuild its stack on day one.

Google's BeyondCorp program, started after the 2009 Aurora attacks and described publicly from 2014 onward, is the canonical large-scale implementation. It removed the corporate VPN, treated every device and user as untrusted by default, and made per-application access decisions based on identity and device posture. The published papers predate the NIST vocabulary but describe recognizably the same architecture, and are a useful next reference for anyone who wants to see a fully realized deployment.

6. The Five Pillars and Your Opsec Program

The CISA Zero Trust Maturity Model v2.0 organizes capabilities into five pillars, with three cross-cutting capabilities that span all of them. Each pillar maps to a domain of a Web3 team's opsec program.

Pillar	Description	Maps to in your program
Identity	Authenticating and authorizing principals	Phishing-resistant MFA, passkeys and hardware keys, SSO, session and token hygiene
Devices	Establishing and maintaining device posture	Device hardening (encryption, updates, screen lock), endpoint baseline, dedicated work devices for critical access
Networks	Segmenting, inspecting, and protecting traffic	Off-chain infrastructure and supply-chain controls, network segmentation, ZTNA where it earns its place
Applications & Workloads	Securing applications and their runtime	Deployment authority, CI/CD and release controls, dependency and supply-chain hygiene
Data	Classifying, protecting, and controlling access to data	Key management and custody, multisig design, secret handling

The three cross-cutting capabilities (Visibility & Analytics, Automation & Orchestration, Governance) span every pillar. They are not separate items because they are not separable practices.

For the Data pillar, note that in Web3 "data" includes private keys, and key protection is the single most consequential application of the posture. Keys are data of the highest sensitivity, so the controls around them should be the most stringent: shortest-lived authentication, tightest least-privilege, hardest assumption of breach.

7. Why This Matters Specifically for Web3

The argument is not abstract. Crypto organizations are high-value, persistent targets. Lazarus Group and other DPRK-aligned APTs run long campaigns specifically against signers, engineers, and finance staff at Web3 organizations. The campaigns last months, the reconnaissance is detailed, the infrastructure for each operation is built fresh. This is a state-resourced industrial process, not opportunistic crime.

The financial logic rests on one fact: a single compromised signer can drain a treasury.

- **Bybit, 2025, $1.5 billion.** Attribution: Lazarus. Vector: a Safe{Wallet} frontend injection that rewrote the transaction signers approved. The signers followed standard procedure on standard tools; the tools were compromised. The attack landed because the architecture trusted the frontend's representation of what was being signed.
- **Ronin Network, 2022, $625 million.** Attribution: Lazarus. Vector: spear-phishing of a Sky Mavis engineer with a fake job offer. Malware on the device gave the attacker four of nine validator keys; a fifth came from an exploited gas-free RPC node. A 5-of-9 threshold reached through one sustained social-engineering campaign.
- **Radiant Capital, 2024, approximately $50 million.** Attribution: DPRK. Vector: malware on signer devices plus UI injection that hid the true call data. A 3-of-11 multisig bypassed because three signers, on three independently compromised devices, approved a transaction whose true effect was hidden from them.
- **Wintermute, 2022, $160 million.** No social engineering. Vector: weak key derivation from "Profanity" vanity-address generation, which produced predictable private keys an attacker could recover.

The supply-chain surface remains active and growing: malicious npm packages impersonating Web3 libraries, poisoned wallet browser extensions, trojanized installers including fake hardware-wallet "recovery" tools.

What ties these together: perimeter security does not protect you from a signer phished on a "job-interview" call. The signer was authenticated, the device was valid, the credentials were valid. Every check the perimeter model is good at returned "yes." The breach was

inside the perimeter from the moment the signer answered the call. Zero trust gives the organization defense-in-depth for when, not if, a team member is compromised.

7.1. Worked Example: The Bybit Signer Flow Under Zero Trust

Under the perimeter model, the post-incident question is "how did the attacker get inside?" The unsatisfying answer is that they did not. The signers were the legitimate signers, on their own devices, using the Safe interface they were authorized to use. There was no "inside" to breach. The model has no useful question to ask about an event in which no perimeter was crossed.

Under zero trust the questions are different and more productive:

- **Verify explicitly:** was the call data displayed to the signer the same call data the wallet would actually sign? Here, no: the UI lied. A control is an out-of-band display of the actual call data on a device that does not share a software supply chain with the UI. A hardware wallet that decodes and displays call data is one such control; a separate transaction simulator on a different device is another.
- **Least privilege:** did this signing session need to authorize an upgrade that handed full control of the wallet to a new owner, or could the operation have been scoped more narrowly? In a stricter design, signers could approve specific outflows but not ownership transfers, with transfers requiring a separate, slower, more heavily-witnessed ceremony.
- **Assume breach:** if the UI is lying to several signers at once (which it was), what catches the attack before it lands? A delay window, an off-chain second-confirmation channel, an automated post-signing check comparing signed call data to a previously-approved policy.

None of these would have been individually sufficient. The defensive question is not "what one control would have stopped this" but "what combination of controls, each cheap to implement, would together have made the attack uneconomic." That is the zero-trust calculation.

8. Where To Start

You cannot deploy a full PDP/PEP stack on day one, and you do not need to. Zero trust is incrementally deployable by design. Order the work by blast radius: protect the assets whose compromise ends the company first.

1. **Inventory the crown jewels.** List what a single compromise could destroy: treasury Safes, deploy keys, DNS and registrar control, cloud and KMS access, source-control org admin. These are your highest-sensitivity resources. Everything else is secondary.
2. **Fix identity first.** Phishing-resistant MFA (hardware keys or hardware-backed passkeys, not SMS or TOTP) on every account that touches a crown jewel. Identity is the new perimeter; this is the cheapest, highest-impact control.
3. **Remove standing privilege.** No permanent admin on production. Move to just-in-time, time-bounded elevation for deploys, infrastructure, and sensitive SaaS. Even a manual request-and-grant process beats standing rights.
4. **Establish device posture for critical roles.** Signers, deploy-key holders, and admins use hardened, ideally dedicated devices. Encryption, updates, screen lock, no risky extensions, work separated from personal browsing.
5. **Engineer assume-breach into custody.** Multisig with a threshold above one and genuinely independent signers, one hardware wallet each, independent calldata verification, no blind-signing. Hot/cold separation to bound blast radius. This is the Data pillar applied to keys.
6. **Add visibility.** Alert on every treasury transaction, owner/threshold/module change, and privileged access grant. Confirm someone reads the alerts. Detection is the backstop for the controls that fail.
7. **Iterate.** Pick the pillar where you are weakest, raise it one maturity level, repeat. The work is never finished; it is improved as the team, the threats, and the available controls evolve.

The remaining guides in this series specify each of these controls. Read each one as an instance of the same posture: verify explicitly, grant least privilege, assume breach.

9. References

- NIST SP 800-207, Zero Trust Architecture: https://csrc.nist.gov/pubs/sp/800/207/final
- CISA Zero Trust Maturity Model v2.0: https://www.cisa.gov/zero-trust-maturity-model
- Google BeyondCorp: https://cloud.google.com/beyondcorp
- John Kindervag, "No More Chewy Centers" (Forrester, origin of the zero-trust model): https://www.forrester.com/report/No-More-Chewy-Centers-Introducing-The-Zero-Trust-Model-Of-Information-Security/RES56682
- Safe post-mortem on the Bybit signing-interface attack: https://safe.global/blog/recovering-and-rebuilding-after-the-bybit-attack