SETUP GUIDE

Browser and Wallet Hardening Guide for Web3 Teams

Securing the highest-traffic Web3 attack surface: the browser and the signer

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 anyone in a Web3 team who interacts with on-chain systems through a browser, whether you are a developer, an operator, or in a non-technical role. The contracts may be audited and the protocol may be sound, and the loss still happens at the browser and the signer. That is where the value moves and where the human makes the final decision.

Most real losses do not come from a contract bug. They come from a signature: an unbounded `approve`, an off-chain `Permit2` authorization, a typed-data message handed to a drainer, or a transaction whose recipient was swapped by a malicious extension or a poisoned address. The attacker does not need to break the protocol if they can get you to sign.

> **NOTE**
> **Baseline rule:** separate browser profiles by risk tier, keep extensions minimal and vetted, verify every transaction on the signing device, and revoke stale approvals.

For the most sensitive work, treasury signing, contract administration, multisig participation, and release approvals, a dedicated signing device with a hardware wallet and a clean profile is still the better option. A browser-extension wallet holding meaningful value is a standing liability.

2. Browser and Profile Strategy

The problem is bleed. Cookies, sessions, extensions, and cached permissions that belong to one activity become available to another when they share a profile. A malicious extension installed for "degen" experimentation can read your work session. A drainer dApp opened in the same profile as your treasury wallet can prompt that wallet to sign. A phishing page that captured a session cookie now has whatever that profile could reach.

Separation by risk tier breaks these chains. Each tier gets its own profile (or, for the highest tier, its own browser or device), its own extension set, and its own wallet. Nothing crosses.

Tier	Purpose	Wallet	Extensions	Notes
Signing	Treasury, multisig, contract admin, high-value transactions.	Hardware wallet only. No extension wallet holding value.	Minimum needed to sign. Ideally a dedicated browser or device.	Never used for general browsing, email, chat, or links from anywhere.
Work	Source control, cloud, docs, dApp interaction with the hot operational wallet.	Hot wallet with capped balances and capped approvals.	Vetted, work-required only.	No personal browsing, no airdrops, no random dApps.
Personal / degen	Experimentation, airdrops, new dApps, mints, trading.	Burner or low-value wallet. Nothing you cannot lose.	Treat anything here as compromisable.	Separate profile or separate browser. Never touches work or signing sessions.

Browser profiles are an isolation boundary for sessions and extensions, not a hardened sandbox. Treat them as risk separation, not as a guarantee that a compromised personal profile cannot affect the machine. For the signing tier, prefer a separate browser or a separate device over a profile in the same browser.

3. Quick Setup Checklist

Every browser used for Web3 work should have these controls before it touches a wallet.

Browser Settings

Setting	What To Do	Notes
Updates	Keep the browser on automatic updates.	Renderer and extension-API bugs are patched often. Do not sit on an old build.
Profiles	Create separate profiles per risk tier.	Signing, work, and personal/degen at minimum.
Sync	Do not sync the signing or work profile into a personal account.	Synced extensions and sessions defeat tier separation.
Default search	Avoid clicking search results or ads for dApps.	Fake dApp ads are a standard delivery channel for drainers.
Site permissions	Deny clipboard, camera, and notification access by default.	Grant only per-site when there is a reason.

Profiles and Wallets

Area	What To Do
Signing profile	Hardware wallet only. No extension wallet holding value. No general browsing.
Work profile	Hot operational wallet with capped balances and capped approvals.
Personal / degen	Burner wallet, separate profile or browser, nothing of value.
Seed isolation	Never import a hardware-wallet seed into an extension wallet.

Extension Review

Area	What To Do
Inventory	List every extension in every work and signing profile.
Removal	Remove anything not actively needed.
Permissions	Scrutinize any extension that can read or change page content or intercept requests.
Source	Install only from the official store, verified publisher, with a real review history.

Allowance Review

Area	What To Do
Stale approvals	Review token and NFT approvals monthly on Revoke.cash or an equivalent.
Unbounded approvals	Revoke or reduce infinite `approve` and broad `Permit2` allowances.
Disconnect	Disconnect dApps and WalletConnect sessions you are no longer using.

4. Browser Extensions: The Risk Surface

A browser extension with content-script access can read everything on a page, rewrite the DOM, intercept and modify network requests, and inject scripts. An extension with that access sitting in your wallet's profile can swap a recipient address before you see it, rewrite a displayed balance, replace a QR code, or relay your session to an attacker. The extension does not need to be the wallet to attack the wallet.

Vet before installing, and re-vet periodically because extensions update silently and ownership changes hands:

- **Publisher.** Known developer, verified where the store supports it, consistent identity across the listing and the project's real site.
- **Permissions.** "Read and change all your data on all websites" is the maximum-risk grant. Justify it or do not install.
- **Reviews and history.** Real review volume over time, not a burst of five-star reviews on a new listing.
- **Install source.** Official store only. Never from a search ad, a social post, a chat link, or a "quick fix" instruction in a support DM.

Wallet extensions carry their own risk. A look-alike wallet in the store, a typosquatted name, or a cloned listing with a slightly different publisher is a common delivery method for seed-phrase theft. Install the wallet from the link on the project's verified site, confirm the publisher, and check the user count and review history. A wallet extension with a few hundred users and a recent publish date is a red flag.

"Quick fix" extensions are a recurring trap: a page or a support agent tells you to install an extension to resolve an error, claim an airdrop, or connect a wallet. Treat any unsolicited install instruction as hostile.

Permission Category	What It Allows	Why It Matters
Read and change all data on all sites	Full DOM read/write, script injection on every page.	Can rewrite addresses, balances, and transaction

		details before you sign.
Web request interception	Read and modify network traffic.	Acts as a man-in-the-middle between the dApp and the RPC.
Clipboard access	Read and write the clipboard.	Clipboard hijacking: swaps a pasted address for the attacker's.
Scripting / executeScript	Inject arbitrary code into pages.	Same effect as full content access, applied on demand.
Native messaging	Communicate with a local host application.	Bridges the browser to local processes; expands the blast radius beyond the tab.

5. Wallet Setup and Connection Hygiene

A hot wallet (private key in software, in the browser extension) is always online and always one malicious signature away from drain. A hardware wallet keeps the key in a separate device and forces a physical confirmation. The two are not interchangeable. Hot wallets are for small operational balances you can afford to lose. Hardware wallets are for anything you cannot.

Separate wallets by tier and do not let them merge:

- **Hot operational wallet.** In the work profile. Capped balance. Capped approvals. Used for routine, lower-value interactions.
- **Cold treasury wallet.** Hardware wallet, ideally behind a multisig. In the signing profile or on a dedicated device. Used only for deliberate, verified transactions.
- **Burner.** In the personal/degen profile. Nothing of value. Assume it will eventually be drained.

Connection hygiene matters as much as the wallet itself:

- **WalletConnect and dApp sessions.** Each active connection is standing authorization. Review connected dApps regularly and disconnect anything you are not actively using. A forgotten session is an open door.
- **Custom RPC and networks.** A malicious or hijacked RPC endpoint can feed your wallet false balances, false simulation results, and false transaction outcomes. Add networks only from the project's verified source, not from a pop-up that offers to "add network" for you.
- **Spending caps.** Keep operational balances low. A hot wallet with a small balance limits the loss when, not if, something goes wrong.

Never import a hardware-wallet seed phrase into a browser-extension wallet. Doing so takes the key out of the secure element and puts it in software, discarding the entire reason for the hardware wallet. If a process asks you to "sync" or "import" a hardware seed into an extension, it is either a misunderstanding or an attack.

6. Transaction Signing Hygiene

The signing step is the last line of defense, and it is where most losses are finalized. Blind signing, approving a transaction or message whose contents your device cannot decode and display, means you are trusting the dApp UI completely. The dApp UI is exactly what an attacker controls. Clear signing, where the hardware device decodes and displays what you are actually authorizing, removes that trust dependency. Use wallets and apps that support it, and verify on the device screen, not in the browser.

Specific signature and approval types carry specific risk:

- **`eth_sign`.** Signs an arbitrary 32-byte hash with no human-readable context. Legacy and dangerous. A modern wallet should warn loudly or block it. Never approve an `eth_sign` request you did not initiate and understand.
- **`personal_sign`.** Human-readable message signing. Lower risk for plain login messages, but read the message. A drainer can phrase a malicious authorization as innocuous text.
- **EIP-712 typed data.** Structured, decodable data. The improvement is that it can be displayed clearly. The risk is that users approve it without reading, and `Permit` and `Permit2` authorizations are EIP-712.
- **ERC-20 `approve`.** Grants a spender the right to move your tokens. An unbounded (`type(uint256).max`) approval is a permanent open door until revoked.
- **`Permit` / `Permit2` off-chain signatures.** A single signature grants spending authorization with no on-chain transaction and no gas, so there is no obvious on-chain footprint at signing time. Drainers favor these because the victim never sends a transaction they would scrutinize.

Two mechanical attacks target the moment of signing:

- **Address poisoning.** The attacker sends a dust transaction from an address that shares your counterparty's first and last characters, polluting your history so you copy the wrong address from a transaction list. Always verify the full address, not the truncated display.

- **Clipboard hijacking.** Malware or a malicious extension replaces a copied address with the attacker's at paste time. Verify the pasted address character by character, or use an address book entry you created and verified.

Build the simulate-and-verify habit. Use transaction simulation (Tenderly, the wallet's built-in preview, or a security extension) to see the predicted state changes before signing. Then verify the target and the calldata on the hardware screen. The dApp can show you anything. The hardware device shows you what you are signing.

Signature / Approval Type	What It Authorizes	Risk
`eth_sign`	Signs an arbitrary hash, no context.	Highest. Can authorize anything. Block or treat as hostile.
`personal_sign`	Signs a readable message.	Moderate. Read it. Can be phrased to deceive.
EIP-712 typed data	Signs structured, decodable data.	Moderate to high. Includes `Permit` authorizations. Read fields.
ERC-20 `approve` (bounded)	Spender may move up to N tokens.	Limited to N. Prefer exact amounts.
ERC-20 `approve` (unbounded)	Spender may move all tokens, indefinitely.	High. Open door until revoked.
`Permit` / `Permit2` off-chain	Gasless spending authorization via signature.	High. No on-chain footprint at signing. Drainer favorite.
`setApprovalForAll` (NFT)	Operator may transfer all NFTs in a collection.	High. Drains an entire collection if granted to an attacker.

7. The Recommended Setup Flow

Step 1: Create Profiles by Risk Tier

Create at least three browser profiles or contexts: signing, work, and personal/degen. For the signing tier, prefer a separate browser or a dedicated device. Do not sync these profiles into a shared personal account, which would re-merge the extensions and sessions you just separated.

Step 2: Clean Up Extensions

Inventory every extension in the work and signing profiles. Remove anything not actively needed. For anything that remains, check the permission grant: any extension that can read or change page content or intercept requests is a candidate for removal. The signing profile should carry the absolute minimum.

Step 3: Set Up Wallet Tiers

Place a low-balance hot wallet with capped approvals in the work profile. Keep the treasury on a hardware wallet, ideally behind a multisig, in the signing context. Put only a burner in the personal/degen profile. Confirm balances match the tier: nothing of value outside the signing tier.

Step 4: Pair the Hardware Wallet

Pair the hardware wallet in the signing profile. Verify the device firmware is current and installed from the vendor's official channel. Confirm clear signing is enabled for the chains and contracts you use. Test with a small transaction and confirm the device screen shows the decoded details, not a blind hash.

Step 5: Review dApp Connections and Allowances

Open Revoke.cash (or an equivalent) for each active wallet. Revoke stale approvals, reduce unbounded approvals, and revoke any `setApprovalForAll` and `Permit2` grants you do not recognize. Disconnect WalletConnect and dApp sessions you are not actively using.

Step 6: Build the Simulate-and-Verify Habit

For every non-trivial transaction: simulate it, read the predicted state changes, then verify the target and calldata on the hardware screen before approving. Make this automatic. The habit is the control.

Step 7: Establish Anti-Phishing URL Discipline

Bookmark the real URL for every dApp and tool you use, and reach them only through the bookmark. Never navigate to a dApp through a search result, an ad, a DM link, or a chat link. Confirm the domain character by character before connecting a wallet. Most drainer campaigns start with a look-alike domain delivered through one of those channels.

8. Extra Rules for Critical-Access Users

Treasury signers, multisig participants, and contract administrators can turn one bad signature into a company-level loss. If this applies to you, the stricter controls below are not optional.

Control	Requirement
Dedicated signing context	Use a dedicated browser or device for signing. No general browsing, email, chat, or external links in it.
Hardware wallet only	No extension wallet holding value. The signing key stays in the secure element.
No seed import	Never import the hardware seed into software, an extension, or a backup app.
Clear signing	Enable and require clear signing. Refuse to blind-sign treasury or admin transactions.
Simulate every transaction	Simulate and read state changes before every signature, without exception.
Verify on device	Confirm target and calldata on the hardware screen, never on the dApp UI alone.
Extension lockdown	No unapproved extensions in the signing context. Re-vet the minimal set periodically.
Allowance discipline	Use exact-amount approvals. Revoke promptly. No standing unbounded grants.
Multisig	Hold treasury behind a multisig so a single compromised signer is not sufficient.
Connection hygiene	Maintain zero idle dApp or WalletConnect sessions in the signing context.
Suspicious activity	Stop signing and escalate before resuming if anything looks off.

9. Things To Avoid

These habits are common, and each one has drained wallets.

- Reaching a dApp through a search ad, a search result, or a link in a DM, chat, or email.
- Importing a hardware-wallet seed phrase into a browser-extension wallet or any software.
- Granting infinite or unbounded ERC-20 approvals when an exact amount would do.
- Signing `eth_sign` requests, or any message or typed data you have not read and understood.
- Approving `setApprovalForAll` or `Permit2` grants to a contract you cannot identify.
- Reusing the signing profile for general browsing, email, chat, or external links.
- Trusting the dApp UI for transaction details instead of verifying on the hardware screen.
- Pasting a recipient address without verifying the full string character by character.
- Copying a counterparty address from transaction history instead of a verified source (address poisoning).
- Adding a custom RPC or network from a pop-up that offers to configure it for you.
- Installing a "quick fix," airdrop-claim, or wallet-connect extension on someone's instruction.
- Leaving dApp and WalletConnect sessions connected long after you finished using them.
- Holding meaningful balances in a hot extension wallet because it is convenient.

10. If Something Suspicious Happens

If you suspect you signed a drainer message, approved a malicious allowance, or connected to a phishing dApp, move fast and in order. Approvals and signatures can be exploited at any time after they are granted, so the window matters.

Immediate Actions

1. Stop signing. Do not approve anything else, including transactions that claim to "fix" or "secure" the wallet.
2. From a clean device or profile, check the affected wallet's approvals on Revoke.cash or an equivalent.
3. Identify what you authorized: an `approve`, a `Permit2` grant, a `setApprovalForAll`, or a session.
4. Notify the security owner or treasury lead from a trusted channel.

Contain

Surface	Action
Token / NFT approvals	Revoke malicious and unbounded approvals immediately, highest-value tokens first.
Permit / Permit2	Revoke or invalidate outstanding off-chain authorizations the wallet granted.
dApp / WalletConnect sessions	Disconnect every active session for the affected wallet.
Assets in the hot wallet	Move remaining assets to a clean, unaffected wallet. Beat the attacker to it.
Suspicious extension	Remove the extension and review what else it could reach.

If a `Permit2` or unbounded approval was granted and assets remain, assume the attacker can move them on their schedule. Revoking the approval is the priority; moving assets out is the backstop if revocation is uncertain or contested by the attacker's transactions.

Rotate and Escalate

- Treat the affected hot wallet as burned. Stop using it for value.
- Generate fresh wallets from a clean device for ongoing operations.
- If a seed phrase may have been exposed (entered into a phishing page, imported into a malicious app, stored insecurely), the wallet and every account derived from that seed are compromised. Move everything and retire the seed.
- Review the profile and device the signing happened on. If a malicious extension or page was involved, assume the profile is compromised and rebuild it.
- For treasury or multisig exposure, escalate immediately and freeze further signing until the security owner agrees it is safe to resume.

11. Monthly Self-Check

Once a month, do this short review. It should take a few minutes.

Check	Done
Browser is up to date.	
Profiles are separated by risk tier (signing, work, personal).	
The signing context is not used for general browsing or links.	
Extensions in work and signing profiles have been reviewed.	
No extension has unjustified read/change-all-sites or request-interception access.	
Wallet tiers are correct: nothing of value outside the signing tier.	
No hardware-wallet seed has been imported into software.	
Token and NFT approvals have been reviewed on Revoke.cash or an equivalent.	
Unbounded approvals and unknown `Permit2` / `setApprovalForAll` grants have been revoked.	
Idle dApp and WalletConnect sessions have been disconnected.	
Clear signing is enabled and used for signing-tier transactions.	
dApp URLs are reached through verified bookmarks, not search or links.	

12. References

- Ethereum signing methods and `eth_sign` risk: https://docs.metamask.io/wallet/concepts/signing-methods/
- EIP-712 typed structured data: https://eips.ethereum.org/EIPS/eip-712
- EIP-2612 permit (ERC-20 signed approvals): https://eips.ethereum.org/EIPS/eip-2612
- Uniswap Permit2: https://docs.uniswap.org/contracts/permit2/overview
- Revoke.cash (approval review and revocation): https://revoke.cash/
- Revoke.cash learn (approvals, exploits, address poisoning): https://revoke.cash/learn
- MetaMask security best practices: https://support.metamask.io/privacy-and-security/staying-safe-in-web3/
- MetaMask on address poisoning: https://support.metamask.io/privacy-and-security/staying-safe-in-web3/what-is-an-address-poisoning-scam/
- Tenderly transaction simulator: https://tenderly.co/transaction-simulator
- Ledger clear signing initiative: https://www.ledger.com/clear-signing
- Chrome Web Store extension permission warnings: https://support.google.com/chrome_webstore/answer/186213