SETUP GUIDE

Domain, DNS, and Frontend Integrity Setup Guide for Web3 Teams

Protecting the registrar, DNS, and the dApp frontend from hijack

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 the people who administer your domains, DNS, and the dApp frontend: registrar admins, DNS operators, and frontend/hosting engineers. The on-chain contracts can be audited and flawless, and users can still lose funds, because the path between the user and those contracts runs through a domain name, a set of DNS records, a TLS certificate, a CDN, and a JavaScript bundle. An attacker who controls any one of those can swap the contract address, inject a drainer, or proxy every transaction through a malicious approval prompt. The user sees the correct URL and the correct branding, signs, and is drained.

This attack class is real and has caused well-known losses across DeFi. It does not require a smart-contract bug. It requires a hijacked registrar account, a changed nameserver, a dangling CNAME, a compromised deploy token, or one malicious npm dependency in the build.

> **NOTE**
>
> **Baseline rule:** lock the registrar with hardware MFA and a registry/transfer lock, sign your DNS with DNSSEC and constrain certificate issuance with CAA, lock down hosting and deploy credentials, and monitor continuously for hijack.

The aim is not to make the frontend unhackable. The aim is to make the high-impact paths (registrar takeover, nameserver swap, deploy-pipeline compromise) hard to reach and fast to revert, because during a frontend hijack your users are being drained in real time.

2. The Frontend Attack Surface

Treat the frontend delivery chain as a sequence of trust handoffs, each of which is a target. The classes below are the ones that matter in practice.

Attack Class	How It Works	Typical Entry Point
Registrar account takeover	Attacker logs into the registrar and changes nameservers, transfers the domain, or edits records.	Weak or SMS-based MFA, phished credentials, social engineering of registrar support, shared login.
DNS record / nameserver hijack	Records (A, AAAA, CNAME, NS) are changed to point the domain at attacker infrastructure.	Compromised DNS-provider account or API token, registrar takeover, weak provider MFA.
Dangling DNS / subdomain takeover	A CNAME or record still points at a deprovisioned service (cloud bucket, CDN, SaaS host); the attacker re-registers that resource and serves content from your subdomain.	Decommissioned services left in the zone, wildcard records, stale staging hosts.
CDN or edge-worker injection	Attacker modifies content at the edge: a worker, a transform rule, or origin override that rewrites the served bundle.	Compromised CDN account, over-scoped API token, unrestricted edge-worker deploy access.
Compromised build or deploy pipeline	Attacker pushes a malicious build or replaces deployed artifacts using stolen CI/CD or hosting credentials.	Long-lived deploy tokens, secrets in CI logs, compromised maintainer account, no build provenance.
Malicious frontend dependency	A drainer or exfiltration payload is introduced through a compromised or typosquatted npm package pulled into the bundle.	Unpinned dependencies, no review of lockfile changes, postinstall scripts, transitive dependency compromise.

Two observations shape the rest of this guide. First, several of these classes chain: a registrar takeover gives DNS and certificate control, which gives full content control.

Second, the user-visible URL stays correct in every one of them, so users cannot defend themselves. Detection and reversion speed are your main controls.

3. Quick Setup Checklist

These are the controls every production domain and frontend should have before launch.

Registrar

Setting	What To Do	Notes
MFA	Enable hardware security keys (FIDO2) on the registrar account. Remove SMS as a factor.	SMS is SIM-swappable and not acceptable for a domain that fronts funds.
Transfer lock	Set `clientTransferProhibited` (and `clientUpdateProhibited` where supported).	Blocks unauthorized transfers and record changes at the registrar level.
Registry lock	Enable registry-level lock for high-value domains.	Requires out-of-band, manual verification at the registry to make any change.
Account	Use a dedicated organizational registrar account, not a personal one.	Survives staff turnover and is in scope for offboarding.
Auto-renew	Enable auto-renew and verify billing. Confirm WHOIS/contact records are current.	A lapsed domain can be re-registered by anyone.

DNS

Setting	What To Do	Notes
DNSSEC	Sign the zone and publish the DS record at the registrar.	Prevents forged-record and cache-poisoning attacks downstream.
CAA	Publish CAA records naming only the CAs you use.	Constrains which CAs may issue certificates for the domain.
Dangling records	Remove records pointing at deprovisioned services. Audit	Closes subdomain-takeover paths.

	CNAMEs and wildcards.	
Provider access	Hardware MFA on the DNS-provider account; scoped, short-lived API tokens only.	Treat the DNS console as production infrastructure.
TTLs	Use sane TTLs: low enough to revert fast, high enough to be stable.	Long TTLs slow your recovery during a hijack.

Hosting and Deploy

Setting	What To Do	Notes
Deploy credentials	Use OIDC/workload identity instead of long-lived deploy tokens where possible.	Removes a static secret an attacker can steal and reuse.
Immutable deploys	Deploy versioned, immutable artifacts with one-click rollback.	Fast reversion is your primary incident control.
SRI / CSP	Add Subresource Integrity to third-party scripts and a restrictive Content Security Policy.	Limits and detects injected or modified scripts.
Dependencies	Pin and lock frontend dependencies; review every lockfile change.	Closes the malicious-dependency path.
CDN / workers	Restrict who can edit edge workers, transform rules, and origin settings.	Edge config is a content-control surface.

Monitoring

Setting	What To Do	Notes
DNS change alerts	Alert on any change to NS, A/AAAA, CNAME, MX, DS, and CAA records.	First signal of a hijack in progress.
Certificate transparency	Monitor CT logs for unexpected	Catches mis-issuance and CAA

	certificates issued for your domains.	bypass.
Page integrity	Continuously check the served bundle hash / DOM against a known-good baseline.	Detects edge injection and deploy tampering.
Domain expiry	Alert well ahead of registrar expiry.	Prevents accidental lapse.

4. Registrar Hardening

The registrar account is the root of trust for the entire frontend. Whoever controls it controls the nameservers, and through them the certificates and the served content. Treat it as the most sensitive account your organization holds outside treasury signing.

Baseline Rules

- Protect the registrar account with hardware security keys (FIDO2/WebAuthn). Remove SMS and TOTP-only fallbacks where the registrar allows it. SMS is defeated by SIM swap; the registrar that fronts your funds is exactly the account a SIM swap targets.
- Set `clientTransferProhibited` to block transfers. Add `clientUpdateProhibited` and `clientDeleteProhibited` where the registrar supports them. These EPP status codes are set at the registrar and stop the most common takeover actions.
- For high-value domains, enable registry lock. Registry lock sits one level above the registrar: changes require manual, out-of-band verification at the registry operator, so even a full registrar-account compromise cannot silently change your nameservers.
- Use a dedicated organizational registrar account tied to a role mailbox, not an individual's personal account or personal email. Personal accounts walk out the door with the person and are invisible to offboarding.
- Restrict and separate access. Few people should have registrar access. Where the registrar supports roles, separate "view" from "change." Log who has access and review it quarterly.
- Enable auto-renew, verify the payment method does not expire, and keep contact and WHOIS records accurate. A domain that lapses can be registered by anyone, and recovery is slow and uncertain.
- Beware registrar support as an attack path. Social-engineering the registrar's support desk has been used to move domains. Where offered, enable account PINs or phone-support passcodes.

Access Review

Check	Pass Condition

MFA type	Registrar uses hardware keys, not SMS.
Transfer lock	`clientTransferProhibited` is set on production domains.
Registry lock	High-value domains have registry-level lock.
Account ownership	Registrar account is organizational, tied to a role mailbox.
Access list	Only required staff have access; the list was reviewed this quarter.
Renewal	Auto-renew is on and billing is valid.

5. DNS Hardening

DNS is the layer that maps your domain to infrastructure and authorizes certificate issuance. An attacker who edits your zone, or poisons resolution downstream, redirects users without ever touching the registrar.

Baseline Rules

- Sign the zone with DNSSEC and publish the DS record at the registrar. DNSSEC authenticates DNS responses and defeats cache poisoning and forged-record attacks on the resolution path. It does not protect against an attacker who controls your DNS account, so it is necessary but not sufficient.
- Publish CAA records that name only the certificate authorities you actually use (RFC 8659). CAA constrains which CAs may issue certificates for your domain, narrowing the mis-issuance surface. Pair it with CT-log monitoring so you see issuance that should not have happened.
- Minimize records and remove dangling entries. Every CNAME or A record pointing at a deprovisioned cloud bucket, CDN endpoint, or SaaS host is a subdomain-takeover waiting to happen: the attacker re-registers the abandoned resource and serves content from your subdomain. Audit the full zone, including wildcards and old staging hosts.
- Treat the DNS-provider account as production infrastructure. Require hardware MFA. Use scoped, short-lived API tokens rather than account-wide long-lived keys. A token that can edit one zone is far less dangerous than one that can edit all of them.
- Separate zones by risk. Keep the production app's apex and primary subdomains in a zone with the tightest access; keep marketing, docs, and experimental subdomains separate so a low-value mistake cannot touch the high-value records.
- Monitor record changes. Alert on any change to NS, DS, A/AAAA, CNAME, MX, and CAA. A nameserver change you did not make is a hijack in progress.
- Use sane TTLs. Keep TTLs on critical records low enough that you can revert a malicious change quickly, but not so low that normal operation is fragile. During an incident, a 24-hour TTL is 24 hours of attacker-controlled resolution cached at resolvers.

DNS Review

Check	Pass Condition
DNSSEC	Zone is signed and the DS record is published at the registrar.
CAA	CAA records name only the CAs in use.
Dangling records	No record points at a deprovisioned service.
Provider MFA	DNS-provider account uses hardware MFA.
API tokens	Tokens are scoped and short-lived; no account-wide static keys.
Change alerts	Alerts fire on NS, DS, A/AAAA, CNAME, MX, and CAA changes.

6. Frontend Build and Hosting Integrity

The served bundle is the last hop to the user. Integrity here means: only code you reviewed and built reaches the browser, you can prove what was deployed, and you can revert it in seconds.

Build and Deploy

- Replace long-lived deploy tokens with OIDC / workload identity from CI to the hosting provider where possible. A static deploy token is a reusable secret that, once stolen, lets an attacker push a malicious build at will. Short-lived, identity-bound credentials remove that standing capability.
- Establish build provenance. Build in CI from a known commit, record what was built, and (where supported) attach signed provenance attestations. You want to answer "what exact artifact is live, and which commit produced it" from records rather than memory.
- Deploy immutable, versioned artifacts with one-click rollback. The single most useful control during a frontend hijack is reverting to a known-good build immediately. Mutable in-place deploys make that slow and uncertain.
- Lock down the deploy path: protected branches, required review, and restricted membership on the CI project and the hosting account. Treat a compromised maintainer account as in scope.

Content Integrity in the Browser

- Add Subresource Integrity (SRI) hashes to third-party scripts and styles so the browser refuses to execute a modified resource. SRI is most effective for assets served from third parties (analytics, widgets, CDNs).
- Deploy a restrictive Content Security Policy. Constrain `script-src`, `connect-src`, `default-src`, and friends so injected inline scripts and exfiltration endpoints are blocked. A tight CSP turns many injection attempts into console errors instead of drained wallets. Use CSP reporting to detect attempts.
- Pin and lock frontend dependencies, and review every lockfile change as carefully as source. The malicious-dependency path (typosquats, compromised maintainers,

postinstall scripts, transitive compromise) puts attacker code directly in your bundle. Minimize dependencies, audit additions, and be skeptical of postinstall scripts.

CDN, Edge, and Decentralized Hosting

- Restrict CDN and edge-worker access. Edge workers, transform rules, and origin overrides can rewrite the served page after it leaves your origin. Limit who can deploy them, require review, and monitor their configuration.
- For IPFS/ENS-hosted frontends, the trust model shifts but does not vanish. The content hash is immutable, but the mutable pointer (an ENS record, a DNSLink TXT record, or the gateway your users hit) is the takeover target. Protect the ENS controller key like a treasury key, secure the DNSLink record like any other DNS record, and remember that a centralized gateway reintroduces a centralized hijack point.
- Run continuous page-integrity monitoring. Compare the served bundle hash and key DOM elements against a known-good baseline from multiple network vantage points, and alert on drift. This is what catches edge injection and tampered deploys that leave the URL and certificate intact.

7. The Recommended Setup Flow

Step 1: Lock the Registrar

Enable hardware security keys on the registrar account and remove SMS. Set `clientTransferProhibited` (and `clientUpdateProhibited`/`clientDeleteProhibited` where supported). For high-value domains, enable registry lock at the registry. Move the account to an organizational role mailbox if it is not already, and confirm auto-renew and contact records.

This is the highest-impact step, because the registrar controls everything below it.

Step 2: Enable DNSSEC and CAA

Sign the zone with DNSSEC and publish the DS record at the registrar. Publish CAA records naming only the CAs you use. Confirm both from an external resolver, not just the provider console, and start monitoring certificate transparency logs for your domains.

Step 3: Remove Dangling Records and Lock DNS-Provider Access

Audit the full zone. Remove any record pointing at a deprovisioned service, and review wildcards and stale staging hosts. Put hardware MFA on the DNS-provider account, replace account-wide static API keys with scoped short-lived tokens, and separate high-value zones from low-value ones.

Step 4: Lock Hosting and Deploy Credentials

Restrict membership on the CI project and hosting account, enforce protected branches and required review, and move from long-lived deploy tokens to OIDC/workload identity where the provider supports it. Confirm deploys are immutable and versioned with a tested one-click rollback.

Step 5: Add SRI and CSP and Review Frontend Dependencies

Add SRI hashes to third-party scripts and styles. Deploy and tune a restrictive Content Security Policy with reporting enabled. Pin and lock dependencies, audit the current dependency tree, and put lockfile changes under mandatory review.

Step 6: Set Up Integrity and DNS-Change Monitoring with Alerts

Configure alerts on DNS record changes (NS, DS, A/AAAA, CNAME, MX, CAA), CT-log issuance, page-integrity drift, and domain expiry. Route alerts to a channel that is watched out of hours, and test that each alert actually fires.

Step 7: Write the Incident and User-Communication Runbook

Write down, before you need it: who can lock the registrar, who can revert DNS, who can roll back the frontend, and how you warn users out-of-band. Pre-stage the user-warning messages on the channels you control directly (your verified social accounts, status page, in-app banner) so you are not drafting copy while users are being drained. Rehearse it.

8. Extra Rules for Critical-Access Users

Production frontends, the domains that front user funds, warrant stricter controls than a marketing site. If you administer those, use the profile below.

Control	Requirement
Registry lock	Production fund-facing domains use registry-level lock, not just registrar transfer lock.
Dual control	Any registrar or DNS change to a production domain requires a second authorized approver.
Dedicated admin identities	Registrar, DNS, and hosting admin accounts are dedicated, monitored, and not shared or reused for personal services.
Change windows	Make changes to production DNS and registrar settings in announced windows, not ad hoc, so an unexpected change stands out.
Out-of-band verification	Confirm any requested registrar/DNS change through a second channel before executing, especially changes that arrive by email or chat.
Hardware MFA	All three layers (registrar, DNS provider, hosting) use hardware security keys.
Deploy authority	Production deploys use identity-bound short-lived credentials and require review; no standing deploy tokens.
Monitoring coverage	DNS-change, CT-log, and page-integrity alerts are confirmed working for every production domain.
Key custody	ENS controller keys or DNSSEC signing keys for production are held with treasury-grade custody, not on an admin laptop.
Suspicious activity	Treat any unexpected registrar email, DNS change, or certificate issuance as an incident until

	proven otherwise.

9. Things To Avoid

These are common and each one has a direct path to a drained user.

- A registrar account protected only by SMS 2FA. SIM swap defeats it, and the domain that fronts funds is the prime target.
- Running production without DNSSEC or without CAA, leaving resolution forgeable and certificate issuance unconstrained.
- Dangling subdomain CNAMEs pointing at deprovisioned cloud buckets, CDNs, or SaaS hosts, waiting to be claimed.
- Shared registrar or DNS logins, or accounts tied to a personal email that offboarding will never touch.
- Unpinned frontend dependencies, and merging lockfile changes without review.
- Unrestricted CDN or edge-worker access, where anyone with console access can rewrite the served page.
- Long-lived deploy tokens sitting in CI, reusable by anyone who steals them.
- No monitoring on DNS changes, CT logs, or page integrity, so the first signal of a hijack is a user complaint.
- Letting a domain lapse. An expired domain can be re-registered by an attacker, and you may not get it back.
- Trusting the URL bar. The correct URL appears in every hijack class in this guide; it is not evidence the frontend is yours.

10. If Something Suspicious Happens

A frontend hijack is an active-loss event. Unlike a quiet endpoint compromise, every minute the malicious frontend is live, users are signing attacker transactions. Speed beats forensics here. Warn users and cut off the malicious content first; investigate after.

Immediate Actions

1. Warn users out-of-band immediately, on every channel you control directly: verified social accounts, status page, Discord/Telegram announcement, in-app banner if reachable. Tell them to stop signing and to revoke approvals.
2. Consider taking the frontend offline. A maintenance page or a hard "do not use" is safer than a hijacked one that drains users. A site that is down loses no funds.
3. Lock the registrar: confirm transfer lock and registry lock are intact, and check for unauthorized nameserver or contact changes.
4. Revert DNS to known-good records. Low TTLs (from Section 5) determine how fast this propagates.
5. Roll back the frontend to the last known-good immutable build.

Contain and Investigate

Layer	Action
Registrar	Verify lock status, review recent account activity and emails, force re-auth, rotate the account credential and MFA if compromise is plausible.
DNS provider	Audit all records against known-good, revoke all API tokens, rotate provider credentials, review change logs.
Certificates	Check CT logs for certificates you did not request; revoke any unauthorized certificate and tighten CAA.
Hosting / CDN	Revoke deploy tokens and sessions, review edge workers and transform rules, confirm the live artifact matches a reviewed build.
Build pipeline	Inspect recent CI runs and deploys, review lockfile and dependency changes, look for an injected or compromised

	package.
Edge / workers	Remove any worker or rule you did not author; confirm origin settings.

Rotate Everything Exposed

Rotate registrar credentials and MFA, DNS-provider credentials and all API tokens, hosting/CDN deploy credentials and tokens, and any signing key (ENS controller, DNSSEC) if its exposure is plausible. Assume any credential reachable from a compromised account is exposed.

Post-Incident Communication

Once the frontend is confirmed clean and reverted, tell users clearly: what happened, what the safe URL is, what they should do (revoke approvals, check for unauthorized transactions), and when it was resolved. Keep the warning visible until you are certain propagation and caches have cleared. Do not declare ″all clear″ while a long-TTL record may still resolve to attacker infrastructure somewhere.

Do not resume normal operation until the registrar, DNS, certificates, hosting, and build pipeline have each been verified clean and the responsible owners agree.

11. Monthly Self-Check

Once a month, do this short review across all production domains.

Check	Done
Registrar uses hardware MFA; SMS is removed.	
Transfer lock is set; registry lock is active on high-value domains.	
Registrar account is organizational and the access list was reviewed.	
Auto-renew is on and domain expiry is not approaching unnoticed.	
DNSSEC is signed and the DS record is published.	
CAA records name only the CAs in use.	
The zone has no dangling records; wildcards and stale hosts were reviewed.	
DNS-provider account uses hardware MFA and only scoped, short-lived tokens.	
Deploy credentials are short-lived/OIDC; no standing deploy tokens.	
SRI and CSP are in place and CSP reports were reviewed.	
Frontend dependencies are pinned and lockfile changes were reviewed.	
CDN/edge-worker access is restricted and configuration was reviewed.	
DNS-change, CT-log, and page-integrity alerts are confirmed working.	
The incident and user-communication runbook is current and rehearsed.	

12. References

- ICANN, EPP status codes (registrar locks): https://www.icann.org/resources/pages/epp-status-codes-2014-06-16-en
- ICANN SSAC, Registry Lock advisory (SAC040): https://www.icann.org/en/system/files/files/sac-040-en.pdf
- RFC 4033, DNS Security Introduction and Requirements (DNSSEC): https://www.rfc-editor.org/rfc/rfc4033
- ICANN, DNSSEC overview: https://www.icann.org/resources/pages/dnssec-what-is-it-why-important-2019-03-05-en
- RFC 8659, DNS Certification Authority Authorization (CAA): https://www.rfc-editor.org/rfc/rfc8659
- OWASP, Subdomain Takeover (WSTG): https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/10-Test_for_Subdomain_Takeover
- MDN, Subresource Integrity: https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity
- MDN, Content Security Policy (CSP): https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CSP
- Certificate Transparency: https://certificate.transparency.dev/
- Chainalysis on the Curve Finance DNS hijack (frontend/DNS hijack incident): https://www.chainalysis.com/blog/curve-finance-dns-hijack-august-2023/