ONCHAIN DIARY RESEARCH SERIES

Onchain Anti-Scam Yellow Paper

A scam taxonomy, a forensics primer,
and layered defense checklists

Built on the Onchain Diary anti-fraud research series

Version 1.0

August 2026

Onchain Diary × UZEN Labs

theonchaindiary.com · github.com/UZEN-Labs

Contents

Abstract

Onchain scams have industrialized. Mass-registered lookalike domains, templated phishing contracts, and automated laundering paths form a pipeline with a division of labor. What an ordinary user faces is not one clever scammer. It is the pipeline.

This paper condenses the Onchain Diary anti-fraud research series into one document, organized in three layers. First, a **taxonomy**: mainstream onchain scams fall into five classes (approval abuse, signature abuse, contract traps, identity fraud, infrastructure attacks), and each class is explained by mechanism, because defense only holds when it is built on mechanism. Second, **onchain forensics**: how to read the trail yourself with a block explorer — checking approvals, tracing funds, verifying contracts. These skills are due diligence before an incident and the first step of recovery after one. Third, **layered defense**: checklists at three levels, for beginners, active DeFi users, and teams managing other people's funds.

One rule runs through the paper: we state verifiable facts and executable actions, and we do not repeat numbers we cannot verify. All methods rely on public onchain data and public cases.

The Chinese edition of this paper is the primary version; this English edition is its sibling, not a translation of an afterthought. Both ship from the same LaTeX source tree.

1 The threat landscape

1.1 Scams are a factory line

Onchain fraud went from cottage industry to factory in a decade. The work is split: one crew runs acquisition (fake social accounts, airdrop bait, impersonation support), another runs collection (mass-deployed phishing contracts and drainer infrastructure), a third runs laundering (mixers, bridges, layered hops). Once a user fails at any single link, funds enter the line and scatter across dozens of addresses within minutes.

Two consequences follow. First, seeing through one script does not end the threat — scripts are consumables. Second, the defense problem is not "spot the scammer" (their surface identity changes constantly) but "guard the exits." There are exactly two exits by which value leaves a wallet: **approvals** and **signatures**. This paper is organized around those two exits.

1.2 Why internet-era instincts fail here

Three properties break the safety intuitions people carry from traditional finance:

- **Transactions are final.** Bank transfers have recall windows and freezing procedures.

A confirmed onchain transaction is settled. There is no support desk and no appeals counter.

- **Approvals are persistent.** An `approve` is not a payment. It is a standing power of attorney with an amount cap. An approval signed today can still be called next year.
- **Addresses carry no identity.** Receiving addresses are free to generate, unlimited in supply, and disposable. Post-hoc accountability is close to engineering-impossible; the output of onchain forensics is evidence, not a refund.

These three properties push the center of gravity of onchain security forward in time: the five minutes before the funds move matter more than everything after.

2 A taxonomy of onchain scams

The point of a taxonomy is convergence. Scripts vary without limit; mechanisms do not. Read the mechanism and a new script is an old trick in new clothes.

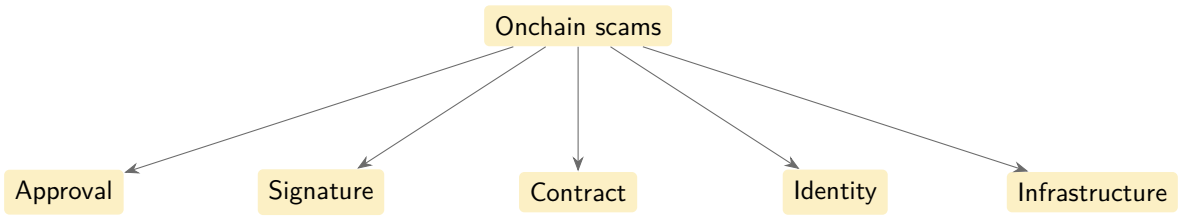

Figure 1: Five classes. Each class attacks one link in the chain between intent and settled value.

2.1 Approval abuse: a blank power of attorney

ERC-20 `approve` grants a contract the right to spend tokens later, up to a cap. Two properties make it dangerous. Most dApps request **unlimited approval** by default — it saves the user one signature and one gas fee on every later interaction. And approvals never expire on their own. An unlimited approval signed for an exchange three months ago is still valid after that exchange exits or gets hacked.

Common entry points: a cloned dApp front-end that induces an `approve` to a hostile contract; a malicious airdrop token whose "claim site" is the approval itself. The countermeasures are as direct as the attack: read the amount field before signing (a raw `uint256.max` is the tell), and sweep unused approvals with a revocation tool on a schedule.

The Permit family pushes this one step further. `permit` and `permit2` complete an approval with a **signature** instead of a transaction — no gas from the user, no visible activity on the

user's address. The attacker pays gas and submits the approval after harvesting the signature. We return to this under signature abuse.

2.2 Signature abuse: you are not signing what you think

The wallet's rendering layer is where information dies. `eth_sign`, `personal_sign`, and typed-data signing all collapse into a few summary lines, and users routinely read "log in" where the message says "transfer" or "approve." Signature scams are all variations on manufacturing that confusion:

- **Fake login.** The site pops a `personal_sign` labeled "sign to accept terms," with a transaction or an approval hidden in the message body.
- **SetApprovalForAll phishing.** The signature grants full NFT collection rights. One signature, every NFT in the wallet becomes moveable. This is the workhorse of NFT drainers.
- **Permit2 phishing.** As above: the signature is the approval, gas paid by the attacker.
- **`eth_sign` blind signing.** The legacy interface signs an arbitrary hash with no displayed content. Legitimate dApps have dropped it; a site asking for it has already told you what it is.

One rule covers the class: **read every character of what the wallet displays before signing**. If you cannot find official documentation explaining the signature, do not sign. "I'll look at it later" is not available, because signatures cannot be revoked.

2.3 Contract traps: the trap is in the code, and the code is readable

Honeypots: buying works, selling is blocked in code. Implementations include seller whitelists (only the deployer's address can sell), dynamic taxes (sell tax jumps to 99%), balance manipulation, and time locks on sells. The onchain signals are clean: a transaction history of buys with no sells, unverified source, or verified source padded with irrelevant logic.

Rug pulls: three shapes — liquidity withdrawal (deployer holds most of the LP unlocked), hidden mint authority (the deployer can mint at will and crush the price), and sell restrictions (the honeypot again). All three are checkable onchain: LP lock status, the contract's mint function and owner powers, holder concentration.

Exploit-driven losses: reentrancy, flash-loan price oracle manipulation, missing access control. Here the first-order victim is usually a protocol, not a user; the user's defense position is protocol due diligence, covered in chapter 3.

2.4 Identity fraud: skip the wallet, attack the person

Fake airdrops: a "check eligibility" link on social media, domain one character off the official one (a hyphen for a dot, an `i` for an `l`, an extra `s`), page pixel-identical to the real site, and a claim button that pops an approval or transfer.

Address poisoning: the attacker generates lookalike addresses sharing your counterparty's first and last characters, then seeds your transaction history with small transfers from them. Later, copying an address out of history grabs the poisoned one. The zero-transfer variant costs the attacker nothing and only needs the record to appear. Defense is unglamorous and works: compare at least six leading and six trailing characters before sending, and test-route large amounts with a small amount first.

Fake support and recovery scams: a user posts publicly about being robbed; "support agents" arrive in DMs within the hour and walk them into a "verification site" that wants the seed phrase. Recovery scams are the second harvest: search "recover stolen crypto" and the results include services that charge an upfront fee and vanish. No legitimate flow ever asks for a seed phrase. That single fact filters most identity fraud.

2.5 Infrastructure: links you do not control

Bridge exploits: bridges concentrate custodied assets and have failed repeatedly at the verification layer. The user-side mitigation is dispersion — do not park funds on a bridge, cross and move on.

Clipboard malware: watches the clipboard, swaps any address-shaped string. Same defense as poisoning: verify after pasting.

Front-end compromise: a dApp's web frontend is hijacked or its domain lapses; the contract is fine, but users are guided into hostile signatures. The countermeasure lives in chapter 4 as environment discipline: verify domains before meaningful actions, distrust abrupt changes in a familiar flow.

3 Onchain forensics: read the trail yourself

Three skills: check approvals, trace funds, verify contracts. The tool is a block explorer (Etherscan and its per-chain equivalents), free, no API key. Before an incident these are due diligence; after one, they are the first hour of recovery.

3.1 Approvals: your list of standing grants

Open your address on the explorer, go to Token Approvals. Every approval you have ever signed is there: the spender contract, the token, the cap, the date. That list is your exposure surface.

How to read it: newest first, and for any spender you cannot place, look it up with the contract method below. Any approval with an unlimited cap gets a keep-or-revoke decision. Revocation costs gas; clearing the backlog in one sitting is cheaper than trickling it.

3.2 Funds: follow the money

If a transfer has already happened, preserve the transaction hash first. The transaction page shows the receiving address, amount, timestamp, and the method called. Open the receiving address and read three things: holdings (ERC-20 and NFT tabs), counterparties (where it went next), and labels (community-maintained tags for exchanges, mixer entrances, known drainer infrastructure).

The laundering pattern is layered: split into dozens of small hops, converge into a mixer or a bridge, exit on another chain. Your realistic job is to archive the path — screenshots and hashes — not because you can claw funds back, but because a complete evidence chain is what police reports and exchange compliance desks act on. If funds land on a centralized exchange deposit address, exchanges have frozen accounts on a well-formed evidence chain. That is the highest-probability recovery path, and it is the entire reason to keep records.

3.3 Contracts: a four-minute check

Given a contract address, read four things in order:

1. **Is the source verified?** The Contract tab with a green check is the floor. An unverified contract offers nothing to evaluate; stop there.
2. **What can the owner do?** Read the code for `owner` powers: mint, pause, change fees. Searching function names in verified source is enough.
3. **Who holds it?** Holders tab: what fraction sits with the top addresses and with the deployer's cluster. Concentration means someone else holds the exit.
4. **Where is the liquidity?** For a trading pair: is the LP locked, and for how long. Unlocked LP can be pulled at any moment.

This does not catch everything — real vulnerabilities need a professional audit. It catches the loud stuff: honeypots, unlocked liquidity, hidden mints all surface in these four checks.

3.4 Address labels: a free identity file

Explorer tags and risk-scoring labels (exchange hot wallets, mixers, drainer clusters, sanctions exposure) are free and continuously updated signals. If the deployer behind a "community airdrop" has historical ties to drainer infrastructure, that outranks anything the website says. Address risk scoring is not magic; it is a weighted read of exactly these trails.

4 Layered defense

Defense is not a pile of tips. It is a stack: base habits everyone needs, and above them progressively stricter operating discipline.

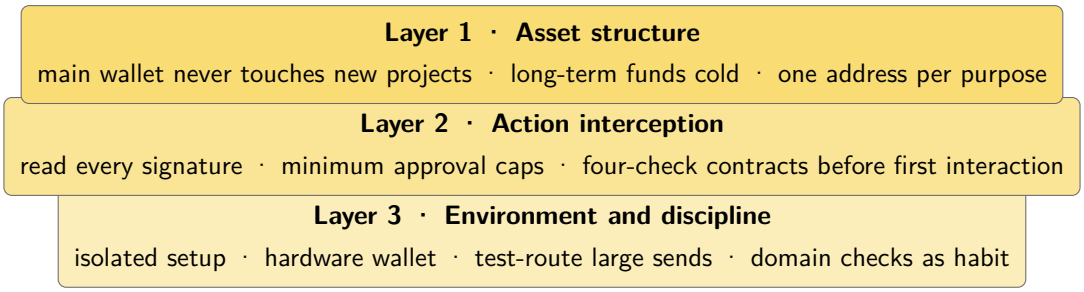

Figure 2: Three layers of defense. When a lower layer fails, the layer above it catches the mistake.

4.1 Layer 1: asset structure

The most consequential decision happens before any scam exists: where the assets sit.

Cold/hot split. Long-term holdings live in cold storage (a hardware device, or offline signing). Daily interaction happens from a hot wallet. The hot wallet balance is your maximum loss in any single incident — set it to a number you can absorb.

Purpose isolation. One address for airdrop farming, one for trading, one for trying new protocols. When one address is phished, the blast radius is that address; triage and cleanup stay simple.

4.2 Layer 2: action interception

There are two exits; this layer guards both.

Signature discipline: read every character displayed; refuse `eth_sign`; if you cannot tell "login" from "approve," do not sign; if the project's own docs do not explain the signature, do not sign.

Approval discipline: choose a capped amount whenever the option exists; revoke what you no longer use — treat it as quarterly cleaning.

Contract discipline: run the four checks from chapter 3 before first interaction with any new protocol. Four minutes against most of the loud traps.

4.3 Layer 3: environment and discipline

Domain habits: enter sites from bookmarks, not from search results or DM links; verify spelling before meaningful actions. This one habit displaces most fake-airdrop and lookalike-domain traffic.

Seed phrase discipline: the seed phrase exists on paper or metal only. Not on any connected device, not in screenshots, not in cloud notes, not in any "verification portal." Any flow requesting it is a scam, without exception.

Transfer habits: first send to a new address is a small test send; after pasting, compare six leading and six trailing characters.

Information discipline: do not broadcast your holdings; treat inbound "support" DMs as hostile by default; treat urgency ("limited," "closes in an hour") as a property of the script, not of the opportunity.

4.4 Institutional addendum

Teams managing other people's funds (DAO treasuries, funds, protocol teams) need, above layers 1–3: multisig as the mandatory asset exit, with single-key authority minimized; written process (who proposes, who approves, how long the timelock runs); and a whitelist for contract interactions, with audits or multiple independent reviews before anything new goes live. Several institutional losses happened not because there was no multisig, but because process was bypassed under time pressure. A procedure is worth what its worst-day enforcement is worth.

5 Case patterns

This chapter names no specific incidents and quotes no unverifiable loss figures. It records patterns. Patterns transfer; numbers do not.

5.1 Pattern one: the eligibility-check chain

A user sees a "legacy-user airdrop check" link on social media. The domain differs from a well-known project's by one character. The page is a complete replica. Connect wallet; "check eligibility" pops a `personal_sign` that reads like a login. The message is a Permit2 approval.

The attacker submits it at their own gas cost; nothing visible happens on the user's address. Days later, the tokens move.

The user had three exits: the entry (bookmark to the official site), the signature (read in full — Permit2 shows up as an allowance message, not a login), and the approval surface (a quarterly Token Approvals sweep would have surfaced an unrecognized grant).

5.2 Pattern two: fake support, second harvest

A user posts publicly about a theft. Within an hour, three "security teams" appear in DMs claiming they can recover funds. One walks the user to an "on-chain verification portal" that wants the seed phrase "to generate a tracking report." The second wallet drains too.

The lesson is one hard rule: the seed phrase has no legitimate input field. Unsolicited helpers are adversaries until proven otherwise.

5.3 Pattern three: the honeypot cycle

A meme token launches to loud community fanfare and a steep chart. Early buyers can sell — that is the bait. As new money arrives, sells start failing: the tax function is switched on. The deployer then pulls the unlocked LP. The onchain signals existed from day one: a deployer-controlled tax switch in verified source, unlocked LP, holder concentration in linked addresses. Not enough people spent four minutes on the checks in chapter 3.

5.4 Pattern four: poisoning lands the copy-paste

A user habitually copies receiving addresses out of transaction history. An attacker has seeded that history with a small transfer from a lookalike address — four matching characters at each end. On a large send, the wrong address gets copied. Finality does the rest.

Six-and-six character checks, or a small test send, would each independently have stopped this.

5.5 What the patterns share

Four entry points (airdrop, support, token, transfer) reduce to a small set of actions: bookmark entry, full-read signatures, four-check contracts, address verification, zero seed-phrase inputs. That is the taxonomy doing its job — scripts change, exits do not, and neither does the defense posture.

6 Appendix A: core terms

Selected from the Onchain Diary glossary (220 entries in full: `https://theonchaindiary.com/glossary/`).

Approval

The ERC-20 operation that lets a contract spend your tokens up to a cap. Persistent: it does not expire on its own.

Unlimited approval

Setting the cap to the maximum. Saves gas, hands the counterparty a blank check.

Permit / Permit2

Approval by signature instead of transaction. Gasless for the user; when abused, "signature equals authorization," and the grant never appears in the user's transaction list.

Blind signing

Signing content the wallet cannot decode. `eth_sign` is the classic case. Legitimate dApps have dropped it; encountering it is a red flag by itself.

SetApprovalForAll

Blanket NFT approval. One signature transfers rights over an entire collection; the backbone of NFT phishing.

Wallet drainer

Phishing-as-a-service: hosted fake sites, signature lures, and asset-extraction infrastructure, rented to scammers for a revenue cut.

Honeypot

A token you can buy but cannot sell. Implemented via seller whitelists, dynamic taxes, balance manipulation, or time locks.

Rug pull

The project team cashing out: liquidity withdrawal, mint-and-dump, sell restrictions, or a combination.

Liquidity lock

LP tokens held by a timelock contract. Unlocked LP can be pulled by the deployer at any moment.

Address poisoning

Seeding a target's history with lookalike addresses to corrupt copy-paste. The zero-transfer variant only leaves the record.

Mixer

A service that breaks the link between funds and their source. A common laundering hop; most are under sanctions scrutiny.

Reentrancy

A bug class where a contract is re-entered mid-call. The direct cause of several protocol exploits.

Flash loan

Uncollateralized borrowing repaid within one block. A neutral DeFi primitive that gives attackers manipulation capital at zero upfront cost.

Sandwich attack

An MEV pattern: a bot brackets your transaction with two of its own and harvests the price impact you caused.

Seed phrase

The wallet's root credential. Any site or service requesting it is a scam, without exception.

Multisig

An address controlled by several keys. The standard exit point for institutional funds.

Timelock

A contract mechanism that delays sensitive actions for a fixed period, giving the community a reaction window.

Address label

An identity tag on an address (exchange, mixer, drainer cluster) maintained by explorers and scoring services. Free, continuously updated.

7 Appendix B: quick-reference checklists

Print-and-pin edition. Each item cites the chapter that explains it.

Before every signature

- ☐ Read every character the wallet displays
- ☐ Cannot explain the signature → do not sign (§2)
- ☐ `eth_sign` → refuse outright (§2)

Before every approval

- ☐ Choose a cap when the option exists (§2)

☐ Treat Permit / Permit2 signatures as approvals, not logins (§2)

Before touching a new token or protocol (four checks, four minutes)

☐ Verified source code (§3)

☐ Owner powers: mint / pause / tax (§3)

☐ Holder concentration (§3)

☐ LP lock status and duration (§3)

Before every transfer

☐ After pasting, compare six leading and six trailing characters (§2)

☐ New address or large amount → small test send first (§4)

Quarterly

☐ Sweep Token Approvals: revoke anything unrecognized (§3)

☐ Re-check asset structure: is the hot wallet still inside your tolerance (§4)

Always

☐ Seed phrase on paper only; zero connected-device input (§4)

☐ Bookmarks, not search results or DM links (§4)

☐ Inbound "support" DMs are hostile by default (§5)

Sources

The methods and mechanism descriptions in this paper are distilled from the Onchain Diary anti-fraud research series (`https://theonchaindiary.com/`): the phishing panorama, five signature-scam variants, unlimited approval risk, six honeypot signals, three rug-pull shapes, address poisoning, the fake-airdrop supply chain, an onchain analysis primer, and tracing stolen funds — 76 articles and a 220-entry glossary at the time of writing. LaTeX sources and version history: `https://github.com/UZEN-Labs/antifraud-yellowpaper`.

Onchain Anti-Scam Yellow Paper v1.0 · Onchain Diary × UZEN Labs · August 2026

For security education only. Nothing in this paper is investment advice.