THE WEB3 SECURITY PLAYBOOK

Immunefi

PUBLISHED DATE 5/Nov/2025

SOURCE IMMUNEFI

TABLE OF CONTENTS

THE RISE OF WEB3 HAS REDEFINED HOW VALUE IS CREATED AND CONTROLLED

The rise of web3 has redefined how value is created and controlled. Protocols now manage billions in assets while operating independently of banks, intermediaries, or national infrastructure. Their security assumptions are almost entirely self-enforced. Financial logic is programmable. Trust is enforced by code. Access is global and continuous.

This shift has introduced a fundamentally different threat surface. Every contract is exposed. Every dollar is liquid. Every attacker operates in full view. Since 2020, more than $70 billion has been lost to protocol exploits. Entire treasuries have vanished. Markets have collapsed. Governance systems have failed under pressure. The tools of permissionless finance are powerful, but fragile.

The core challenge is structural. In web3, finality cannot be reversed. When code is compromised or authority is abused, there is no recourse. No clearinghouse mediates disputes. No delay halts the damage. No regulator enforces restitution. What happens onchain is permanent, and its consequences unfold in real time.

Immunefi was created to meet this reality. We built an adversarial marketplace where vulnerabilities can be reported, verified, and resolved with urgency. Our mission is to protect the protocols that power web3 by aligning the incentives of the builders who create them with the hackers who break them.

Since launch, Immunefi has:

- Protected more than $25 billion in assets across critical infrastructure
- Enabled over $100 million in payouts to Security Researchers
- Built the largest community of Security Researchers in web3
- Partnered with the top protocols in DeFi, infrastructure, and L1 ecosystems

Security must evolve with the systems it protects. Static defenses cannot keep pace with the adversarial dynamics of web3. This playbook is drawn from real incidents and resolved exploits. It offers operational doctrine for teams committed to survival.

1. ANATOMY OF RISK IN DEFI

Protocols fail in predictable ways. Keys are lost. Governance is hijacked. Liquidity vanishes. Exploits spread across integrations. These are not edge cases. They are recurring patterns in web3.

This section outlines common failure vectors observed across the ecosystem. Understanding them is the first step toward building resilient protocols.

1.1 Custody and Key Management

In web3, control ultimately resolves to private keys. They govern access to smart contracts, multisigs, governance modules, and treasuries. Cryptographic authority is absolute. If a key is compromised, so is the system.

Custody risk is foundational. Weaknesses in key generation, storage, or usage lead directly to catastrophic outcomes. These are not theoretical risks. They are among the most frequent causes of major loss.

Private Key Compromise

Private keys are the root of trust. Exposure nullifies every other security measure. Systems fail even if only one key is lost. In traditional finance, fraud may be reversible. In web3, authority is cryptographic and final.

Hot Wallets and the Price of Convenience

Hot wallets store private keys on internet-connected devices. This makes them accessible, but permanently exposed. A phishing link, compromised browser extension, or remote access trojan can be enough to extract the key[1]. Once compromised, funds are unrecoverable.

For teams managing treasuries or operational balances, using hot wallets introduces structural risk. A single compromised device undermines even the most sophisticated contracts.

Insider Risk and Social Engineering

Key compromise often begins with people, not code. Teams are small and reachable. Attackers impersonate investors or recruiters, waiting for a moment of access. The Lazarus Group used a fake job interview to breach Axie Infinity, a tactic repeated across the industry[2].

Even secure tools cannot protect against deception. When teams are small, risk is concentrated.

Operational Mistakes and Attack Surface

Protocols do not need to be hacked to suffer loss. Many incidents stem from operational errors: unencrypted cloud backups, misplaced seed phrases, credentials stored in GitHub, or unrotated keys after offboarding. These are lapses in discipline, not exploits.

In web3, every team action expands the attack surface. Key management practices are part of the system design.

Designing for Irreversibility

Web3 systems do not offer rollbacks. A lost or exposed key means irrevocable loss of control. Key management must be treated as a design challenge. It matters not only who holds authority, but how it can be shared, revoked, and audited under stress.

Custody defines the boundaries of power and shapes the resilience of the protocol.

1.2 Operational Risks

Even the most secure codebase depends on how it is operated. Many of the worst incidents in web3 have started with weak governance, poor key management, or failed processes. These causes appear across a wide range of protocol failures.

Protocols must be run with discipline. Without it, the guarantees provided by the code can fall apart.

Procedural Gaps and Governance Drift

Informal governance introduces quiet vulnerabilities. Common issues include untracked multisig signers, shared admin keys, and incomplete offboarding. These decisions often seem harmless until the system is placed under stress.

Teams need clear roles, reliable authority checks, and tested paths for escalation. When keyholders leave or a deployment freezes funds, the team must have a plan in place. Without one, the damage grows while decisions stall.

Strong governance requires more than a multisig. It requires clarity about who can act, when action is allowed, and how authority can be removed. Without term limits or

formal offboarding processes, multisigs risk accumulating inactive or unaccountable signers, a point recently underscored during the recent Arbitrum Security Council election.

Human Error and Exploitable Habits

Teams frequently make operational mistakes. These range from deploying test code to sending tokens to the wrong address. In web3, such errors cannot be rolled back.

Attackers understand this dynamic. They build exploits around expected behavior, such as address reuse, interface spoofing, and time-based phishing. Habits that go unchecked become entry points.

Protocols need safeguards for these moments. Automated checks, access control, and simulation environments help ensure that failure does not begin with a mistake.

Insider Collusion and Privileged Access

Anyone with elevated permissions can introduce risk. This includes well-intentioned team members who have the ability to propose upgrades, manage keys, or execute governance votes.

Some of the most damaging incidents in DeFi have involved insiders who acted within their apparent rights. By the time patterns are recognized, the protocol may already be compromised.

Protocols should minimize privileges, maintain audit trails, and rotate access regularly. These steps create resilience without assuming that every actor will always act in good faith.

Operational Weakness as Systemic Risk

A protocol is more than code. It is an ongoing operation that requires structure, discipline, and oversight.

Teams that rely on informal habits or undocumented processes create vulnerabilities that grow over time. Robust protocols invest in structure early and reinforce it consistently.

1.3 Infrastructure Risks

Security failures often stem from what surrounds the contract, not the contract itself. Build tools, developer machines, libraries, and deployment systems all present viable targets. These components extend the attack surface and must be treated as part of the protocol's core infrastructure.

Software Bugs and Dependency Flaws

Protocols rely on external software. This includes wallets, libraries, signing tools, and cryptographic implementations. These dependencies carry assumptions that do not always hold under adversarial conditions.

If a bug goes undetected, the result can be irreversible. In a finality-based system, one failed assumption can result in unrecoverable loss. For teams managing upgrade keys or treasury controls, dependency risk is existential.

Endpoint Compromise and Malware

A compromised laptop undermines even the strongest smart contract. In late 2023, attackers compromised a Bybit employee’s device to access a Safe multisig, resulting in over $4 million stolen. Malware does not need to manipulate the blockchain directly. It can wait for a private key to load into memory or for a user to click a spoofed confirmation.

Developer environments often include shared machines, browser-based wallets, cloud platforms, and multiple third-party tools. Each represents a possible entry point. Once breached, these endpoints allow full control over signing activity.

Hardening these systems is not optional. Endpoint hygiene underpins the entire security model.

CI/CD and Supply Chain Vulnerabilities

Most codebases rely on continuous integration and delivery. Pipelines automate deployment and integrate with multiple third-party services. These systems are rarely isolated and often lack the scrutiny applied to production contracts.

Attackers exploit these connections. They insert malicious packages, steal deploy keys, or manipulate build steps. Once compromised, the damage spreads silently. The affected code appears legitimate and is deployed under trusted credentials[3].

Supply chain risk must be treated as a first-class security concern. Every step in the pipeline should be auditable, verifiable, and access-controlled.

Change as a Constant

Infrastructure does not remain static. Teams rotate keys, switch cloud providers, upgrade tooling, and migrate systems. Each adjustment creates a moment of exposure.

Assumptions made at launch become obsolete unless revisited.[4] Security models that ignore these changes begin to drift. What once protected the protocol may no longer apply.

Teams must treat infrastructure as dynamic. Security practices must keep pace through continuous validation, testing, and review..

1.4 Protocol and Economic Design Risks

Protocols are shaped by incentives, assumptions, and design logic. These foundations must be tested for resilience under stress. Failure to do so introduces vulnerabilities that cannot be patched post-deployment.

Misaligned Incentives and Economic Loops

Protocols shape behavior through incentive structures. Some are explicit, such as liquidity rewards or fee sharing. Others emerge unintentionally, like oracle manipulation or flash loan abuse.

Many of the largest incidents in DeFi history involved attackers using valid transactions to extract value. These incidents were not caused by bugs in code but by economic designs that allowed value extraction through legitimate actions.

Every economic mechanism should be evaluated under adversarial conditions. This includes simulations of rational attackers, edge-case market behavior, and incentives that reward harm. Design must account for profit-motivated threats.

Oracle Manipulation

External price feeds introduce risk. Thin markets, aggregation flaws, or latency can all lead to incorrect pricing. Once an oracle is compromised, lending decisions, liquidations, and asset valuations all become unreliable.

Oracles must be selected and implemented with care. Relying on a single source or using naive price averages creates fragility. Systems should include guardrails for volatility and test performance under manipulated inputs.

An oracle is not a passive data source. It is part of the protocol's trust boundary and must be treated as a potential attack surface.

Governance Attack Surfaces

Governance gives communities control over protocol behavior. It also opens new vectors for abuse. Token votes can be purchased, signers can be replaced, and control mechanisms can be hijacked through proposals.

Well-meaning designs can be weaponized. In some cases, attackers have used governance systems to gain full authority without triggering any alarms. The result appears legitimate even when the outcome is destructive.

Teams must secure governance as tightly as they secure contracts. This includes time delays, multi-layer approvals, signer safeguards, and emergency intervention tools. Without this, control structures become liabilities.

Composability and Cascading Risk

DeFi protocols are highly interconnected. One may rely on another's token logic, pricing oracle, or liquidity pool. These links create systemic dependencies.

Attackers exploit these relationships by targeting weak integrations. A change in one system, such as a modified token transfer behavior or an unstable price feed, can cause downstream failures across multiple protocols.

Composability increases surface area. Teams must understand their dependencies, test failure modes, and isolate risk wherever possible. Integration should be intentional and supported by clear fallback behavior.

1.5 Interdependence Risks

DeFi protocols rarely operate in isolation. They are part of a larger system that includes bridges, oracles, wrapped assets, and other integrated tools. These connections enable new functionality, but they also introduce external risk.

Dependency on Bridges and Wrappers

Bridged assets and cross-chain integrations allow protocols to expand functionality across ecosystems. They also introduce reliance on third-party systems that may not be secure.

When a bridge is compromised, as seen in the Wormhole exploit, where attackers bypassed signature verification to steal over $300 million, every protocol depending on that asset inherits the risk. If bridged tokens are used as collateral, liquidity, or governance power, the damage extends well beyond the original point of failure.

Bridged assets must be treated with caution. Protocols should limit exposure, apply discounts to riskier assets, and implement circuit breakers that can respond if upstream systems are attacked.

Integration Assumptions and Protocol Drift

Integrations depend on external systems behaving consistently. When upstream protocols change behavior, upgrade logic, or shift ownership, downstream dependencies can break in unexpected ways.

These changes often go unnoticed until something fails. A minor update in an upstream token or lending protocol can create conditions that affect liquidation, pricing, or governance behavior.

Protocols should actively monitor the systems they rely on. Integrations must include mechanisms for alerting, rollback, and failure isolation to prevent one change from cascading through the stack.

Systemic Liquidity Fragility

DeFi liquidity is highly concentrated. Many protocols rely on the same pools, routes, or aggregators to enable trading and settlement. This shared reliance creates exposure during stress events.

When a major pool becomes illiquid or compromised, the effects are immediate. Pricing breaks down, transactions fail, and protocols depending on these conditions may stop functioning correctly.[5]

Protocols should prepare for this by testing slippage, monitoring real-time liquidity, and defining fallback paths. Liquidity must be treated as a security consideration, not just a market condition.

Ecosystem-Wide Attack Vectors

Some vulnerabilities are not protocol-specific. They stem from shared standards or behaviors adopted across the ecosystem.[6] Examples include reentrancy risks, unsafe token approvals, or flawed interface assumptions.

When one team discovers a new exploit path, others quickly follow. Attackers replicate effective methods across protocols that share the same weaknesses.

Security teams should treat ecosystem exploits as early warnings. Monitoring industry incidents, updating assumptions, and patching shared code can prevent the same exploit from being used repeatedly.

1.6 Regulatory and Geopolitical Risks

Although web3 systems operate on decentralized infrastructure, they remain subject to legal, regulatory, and geopolitical pressure. These forces can affect access, availability, and continuity of service, regardless of code quality.

Sanctions, Blacklists, and Freezing Powers

Infrastructure that supports decentralized systems often relies on centralized components. Frontends, RPC nodes, and stablecoin issuers may comply with legal orders to block or freeze assets.

Governments have exercised these powers in the past, targeting mixers, bridges, and protocols that rely on centralized support systems.[7] Even if smart contracts remain live, access to them can be restricted indirectly.

Teams must account for these risks by minimizing reliance on chokepoints and choosing infrastructure providers with clear legal protections or redundancy options.

Jurisdictional Exposure of Teams and Infrastructure

Many protocols are governed by people. Those people reside in specific jurisdictions and may be subject to court orders or regulatory enforcement.

When authorities cannot directly target code, they often target individuals. This includes founders, signers, or operators with privileged access to hosting, domains, or infrastructure.

Protocols should map their legal exposure. Key functions like DNS management, cloud storage, and multisig access should be designed to withstand pressure from a single jurisdiction.

Legal Risk from Token Design and Governance

The structure of a token can trigger legal scrutiny. Governance rights, revenue distribution, and voting power may imply investment-like behavior that falls under securities regulation.

Legal classifications vary across countries, and interpretations shift quickly.[8] A token that seems compliant today may be interpreted differently tomorrow.

Teams must approach token design with legal diligence. This includes clear disclosures, jurisdictional segmentation, and avoiding structures that suggest entitlement or control.

Unstable Regulatory Environments

Some jurisdictions lack clear guidance on web3. Others apply existing laws unpredictably. This creates a volatile environment where teams may face enforcement without warning.

Past behavior offers limited protection. Actions that were considered acceptable may later be challenged retroactively.[9]

Protocols should plan for legal change. This includes documenting governance decisions, designing for modularity, and preparing for geographic or structural adjustments as laws evolve.

2. DEFENSIVE DESIGN: PRINCIPLES FOR BUILDING SECURE DEFI SYSTEMS

Security must be embedded in how a protocol is built, not added after launch. Defensive design is a development mindset that prioritizes resilience, assumes hostile conditions, and treats simplicity as a form of protection.

2.1 Minimize Trust Surfaces

Every actor with power introduces risk. Defensive design reduces the number of parties who can influence a system and narrows the scope of what they can do.

Teams should avoid relying on single signers, centralized roles, or unverified behavior. Multisigs, quorum systems, and permission controls all reduce the damage that can result from a compromised actor.

This principle extends beyond contracts. It includes keyholders, governance processes, and infrastructure providers. Each should have the minimum necessary authority.

2.2 Defense-in-Depth

Resilient systems do not rely on a single line of defense. They implement multiple safeguards across technical, operational, and legal layers.

Custody systems should require quorum signatures, real-time monitoring, and location diversity. Infrastructure should include anomaly detection and rollback mechanisms. Legal exposure should be distributed across jurisdictions to limit concentrated risk.

As threats evolve, protocols must continuously validate whether their defense layers still hold.

2.3 Default to Least Authority

No actor should hold more access than required for their role. This includes keyholders, governance agents, and automated systems.

Teams must define roles clearly and apply access controls precisely. Signers should be offboarded cleanly, credentials should rotate regularly, and responsibilities should not drift without review.

Protocols must also assume that some participants will act carelessly or maliciously. Restricting authority limits the damage they can cause.

2.4 Attack Surface Reduction via Simplicity

Complex systems create more opportunities for failure. Each added feature, dependency, or configuration expands the space where something can go wrong.

Critical components should remain lean. Avoiding nested logic, unnecessary integrations, and untested dependencies makes it easier to reason about behavior and harder for attackers to find exploitable edge cases.

Simplicity improves reviewability. Auditors, developers, and users can understand and validate the system more easily when it avoids excess.

2.5 Kill Switches, Circuit Breakers, and Time Delays

Time creates space for defense. Protocols need mechanisms that allow teams or communities to intervene when something goes wrong.

Circuit breakers can block unusual activity. Time delays can slow large transactions or upgrades. Emergency switches can pause protocol functions during high-risk situations.

These tools should never rest with a single actor. They must be transparent, governed, and clearly defined. Their purpose is to contain impact, not introduce hidden control.

3. OPERATIONALIZING SECURITY

Security depends on how teams operate. Audits and tooling play a role, but sustained resilience comes from disciplined governance, clean development practices, real-time monitoring, and clear plans for response.

3.1 Security-Informed Governance

Governance defines who holds power and how decisions are made. If authority is unclear or undocumented, attackers can exploit the gaps.

Protocols must assign clear responsibilities. Keyholders should be formally designated, onboarded with training, and offboarded through full revocation. Every sensitive operation, whether contract upgrade, treasury movement, or emergency pause, should have named owners and explicit procedures.

Teams must prepare for failure. This includes maintaining a tested incident response plan, rehearsing roles under stress, and pre-authorizing specific actions. These measures allow rapid action when issues emerge.[10]

Some teams formalize their governance security posture through dedicated external roles, including Immunefi's vCISO service, which translates operational risk into actionable governance policy and ensures continuity through team transitions.

3.2 Developer Hygiene

The development pipeline is part of the security perimeter. Weakness here undermines everything else.

Teams should treat repositories, CI/CD pipelines, and build systems as high-value targets. Access should be limited and auditable. Dependencies must be reviewed and verified. Build steps should be deterministic and reproducible.

No production environment should allow broad access or unmonitored changes. Compromise in development infrastructure is compromise at the protocol level.

3.3 Real-Time Monitoring, Firewalling, and Threat Detection

Static defenses are not enough. Protocols need to watch for threats as they unfold. Onchain monitoring, as delivered by Immunefi's Magnus platform, continuously scans transaction flows, governance changes, and behavioral anomalies, turning live data into actionable alerts. Mature teams use these signals to detect anomalies early and act before issues escalate.

Monitoring should connect directly to governance and response systems. When something unusual happens, teams need not just visibility but the ability to act.

Threats do not always come from inside the protocol. External events, such as phishing, social engineering, or ecosystem exploits, can reveal emerging risk. Intelligence sharing and monitoring of related systems can close this gap.

Immunefi's AI is trained on the largest dataset of live exploits, bug reports, and security events in crypto. This intelligence layer helps anticipate threat patterns before they materialize, providing a proactive edge in monitoring and response.

3.4 Safe Harbor and Emergency Protocols

The ability to respond quickly determines whether an incident causes limited damage or total loss.

Protocols should define emergency processes before they are needed. This includes secure communications, legal and PR contacts, pre-approved disclosure templates, and escalation paths.

Safe Harbor programs, including those offered by Immunefi, create a formal framework for responsible disclosure and rapid coordination with researchers during live exploit scenarios. Several leading protocols have adopted frameworks supported by external security partners to ensure issues can be addressed in-flight without ambiguity. These programs must be visible, governed, and integrated into operations.

Teams must train for stress. Rehearsals, after-action reviews, and continuous improvement help turn plans into habits. Most successful responses begin before the incident occurs.

4. SECURITY AS ECOSYSTEM INFRASTRUCTURE

Most current security practices remain fragmented, split across audits, researcher coordination, monitoring tools, and manual triage. Immunefi integrates these layers into a single operating model, helping protocols keep pace with attackers who operate at machine speed.

Security must scale alongside protocol adoption. The more open and interconnected the ecosystem becomes, the more protocols depend on shared defenses, provided by unified onchain security platforms like Immunefi, which centralize visibility, threat response, and researcher coordination.This section focuses on how security becomes a persistent function across development, operations, and external collaboration.

4.1 The Role of Audits, Audit Competitions, and Bug Bounty Programs

Audits provide valuable insight, particularly when conducted close to deployment and scoped to economic and governance risks. Engaging third-party platforms that offer integrated Audit, Bug Bounty Program, and Audit Competition services allows protocols to sustain pressure through every phase of development.

Bug bounty programs introduce external pressure that helps sustain live security coverage. Platforms like Immunefi enable structured disclosure and incentivized reporting that reinforce protocol resilience.They allow researchers to test systems under real-world assumptions and incentivize discovery before adversaries strike.

Immunefi delivers continuous visibility through its integrated platform: public bug bounty infrastructure, formal audit services, structured audit competitions, and real-time monitoring. These layers function as a security operating system for the onchain economy, continuously adapting to emerging threats. Audit Competitions apply competitive pressure across diverse researchers and real incentives, helping uncover edge cases and overlooked logic beyond what traditional reviews may catch.

4.2 Safe Deployment Playbook

Every protocol upgrade or deployment introduces risk. Transitions expose teams to configuration errors, dependency issues, and signer availability problems.

A safe deployment checklist should include:

- Final audit approval based on current scope
- Verification of all libraries and integrations
- Confirmation of active and reachable signers
- Testnet trials of upgrades and rollbacks
- Active monitoring and alerting from the moment of deployment
- Predefined pause or rollback mechanisms

These steps create structured defenses against operational exposure. Skipping them increases the chance of failure.

Deployment risk is part of the security model. Teams must treat it with the same rigor as contract development.

5. BUILDING SUSTAINING SECURITY CULTURE

Security depends on how teams operate, not just what they ship. A durable culture shapes habits, reinforces standards, and rewards contributions that strengthen the system.

5.1 Founder-Driven Security Commitment

Founders set priorities. When they invest time in incident planning, allocate security budgets, and engage directly with response efforts, teams follow suit. Their decisions make the difference between minimal compliance and operational resilience.

Security leadership is ongoing. It requires presence, not delegation. When leadership maintains focus, the rest of the organization stays aligned.

5.2 Building Security-Conscious Communities

Every user, contributor, and integration partner is part of a protocol's broader defense. Communities that understand phishing, social engineering, and governance manipulation reduce risk across the board.

Teams must publish guidance, build safe defaults into interfaces, and recognize community members who raise concerns or flag anomalies. Repetition builds awareness. Participation builds defense.

5.3 Incentivizing Security Researcher Contributions

Security Researchers test assumptions that teams overlook. Their findings prevent losses, reduce fragility, and improve live systems. Even without formal roles, they act as core contributors.

Strong programs respond quickly to valid reports, pay bounties on time, and maintain professionalism during disclosure. Some researchers work for incentives. Others are motivated by curiosity or reputation. Effective teams create space for both.

Immunefi supports this with tiered bounties, structured disclosure pipelines, and relationships built on trust. These structures turn outside pressure into inside insight.

6. LOOKING AHEAD: COMPOSABILITY AND ITS CONSEQUENCES

Web3 systems are not built in silos. Each protocol connects to others, forming an interdependent stack of contracts, tokens, bridges, and governance systems. These connections bring power but also create shared risk.

Security in this environment depends on awareness of dependencies and how failures propagate. A problem in one contract can ripple outward, affecting others far beyond its original scope.

Protocols must analyze upstream and downstream systems continuously. They need to monitor for code changes, governance shifts, and liquidity pressures that affect integrated components. Dependencies should be mapped, risks modeled, and response plans rehearsed.

Resilience belongs to the protocols that expect failure in the systems around them and prepare to contain it. They isolate critical logic, monitor economic assumptions, and design for degraded performance when external systems behave unpredictably.

Security is no longer about one protocol holding its ground. It is about many protocols protecting each other through shared diligence and proactive alignment. The more we integrate, the more that discipline matters.

REFERENCES

1. Chainalysis. (2022). Lazarus-linked Axie Infinity exploit: Ronin bridge hack analysis. Chainalysis. https://go.chainalysis.com/ronin-bridge-hack
2. Open Source Security Foundation. (2022, January). OpenSSF 2022 Year in Review: Securing the open source software supply chain. https://openssf.org/blog/2023/01/12/openssf-2022-year-in-review/
3. International Monetary Fund. (2021, October). Crypto ecosystem and financial stability challenges, in Global Financial Stability Report, Chapter 2. https://www.imf.org/en/Publications/GFSR/Issues/2021/10/12/global-financial-stability-report-october-2021
4. Möser, M., Böhme, R., & Breuker, D. (2013). An inquiry into money laundering tools in the Bitcoin ecosystem. In Proceedings of the 2013 APWG eCrime Researchers Summit (pp. 1–14). IEEE. https://maltemoeser.de/paper/money-laundering.pdf
5. U.S. Department of the Treasury. (2022, August 8). Treasury sanctions notorious virtual currency mixer Tornado Cash. https://home.treasury.gov/news/press-releases/jy0916
6. Organisation for Economic Co-operation and Development. (2022). Crypto-Asset Reporting Framework (CARF) and amendments to the Common Reporting Standard (CRS). OECD. https://www.oecd.org/tax/exchange-of-tax-information/crypto-asset-reporting-framework-and-amendments-to-the-common-reporting-standard.htm
7. European Securities and Markets Authority. (2023). Trends, Risks and Vulnerabilities (TRV) Report – 1H 2023. https://www.esma.europa.eu/sites/default/files/library/esma50-165-1442_trv_report_1h_2023_0.pdf

8. Safe (SafeDAO). (2025). February 28th, 2025: Statement by the Safe Ecosystem Foundation. SafeDAO. https://safe.global/blog/safe-ecosystem-foundation-statement