OneForge

A Protocol for Atomic, Trustless Token Issuance on Solana

Res

Founder, Crxcible

crxcible.io

March 2026

Revision History

Version	Date	Notes
0.1	March 2026	Initial public draft

Abstract

Token launches on Solana are structurally vulnerable to exploitation at multiple stages of execution. Existing platforms mitigate this through policy-based controls — configurable locks, administrative promises, and multi-step pipelines — none of which are enforceable at the protocol level. OneForge is a protocol for atomic token issuance, live on Solana mainnet since March 2026 with multiple production launches completed. It eliminates these vulnerabilities through a two-phase pipeline in which the critical token invariants — mint authority revocation, freeze authority exclusion, supply burn, and on-chain metadata creation — are executed within a single atomic `VersionedTransaction` and cannot be partially applied. Liquidity pool seeding follows immediately as a guaranteed second transaction, with idempotent checkpointing ensuring the pipeline either completes in full or can be safely resumed from any stage. Partial execution states, the primary vector for rug pulls and administrative exploitation, are made structurally impossible. Token metadata is pinned permanently to Arweave, ensuring immutability beyond the blockchain layer. This paper describes the OneForge protocol, its threat model, its reference implementation via Crxcible, and its implications for trustless on-chain service delivery on Solana. All claims are verifiable against on-chain transaction records — no trust in the platform, the deployer, or this paper is required.

Contents

1 Introduction

The proliferation of token launches on Solana has produced a well-documented pattern of exploitation. Deployers retain administrative privileges after launch, drain liquidity pools, inflate supply beyond advertised caps, and allow metadata to disappear when centralised hosting fails. These failures share a common root: the token issuance process is executed across multiple transactions, each of which creates a window in which the deployer retains control.

Existing platforms have responded with policy-based controls. Liquidity locks are offered as optional features. Authority revocations are performed as separate post-launch steps. Metadata is stored on centralised servers with assurances of permanence. These measures are insufficient because they are configurable by the same party they are designed to constrain.

OneForge addresses this problem at the architectural level. The critical token invariants — authority revocations, supply burn, and permanent metadata — are encoded into a single atomic transaction that cannot partially execute. Liquidity pool seeding follows as a guaranteed pipeline stage with idempotent checkpointing. There is no intermediate state in which a deployer has minted a token but retains the authority to modify it. There is no post-launch step that can be silently omitted.

This paper describes the OneForge protocol as implemented in Crxcible (crxcible.io), a token launchpad live on Solana mainnet. All claims in this paper are verifiable against on-chain transaction records.

2 The Problem: Structural Vulnerabilities in Token Issuance

Token launches on Solana involve a sequence of operations that, when executed across multiple transactions, create exploitable windows. We identify five primary vulnerability classes.

2.1 Mint Authority Exploitation

The SPL Token program [6] allows a designated mint authority to issue additional tokens after launch. If this authority is not revoked atomically at mint, the deployer retains the ability to inflate supply indefinitely. Many platforms revoke mint authority as a separate post-launch transaction, creating a window in which supply can be inflated between mint and revocation.

2.2 Freeze Authority Exploitation

The freeze authority allows a designated account to freeze any token holder's wallet, preventing them from selling or transferring their tokens. Retention of freeze authority after launch enables targeted suppression of selling activity, a mechanism used to manipulate price.

2.3 Liquidity Pool Drain

When liquidity pool creation and LP token locking occur in separate transactions, the deployer temporarily holds LP tokens between steps. During this window, the deployer can remove liquidity before the lock transaction is submitted, resulting in a complete drain of pool liquidity.

2.4 Metadata Impermanence

Token metadata stored on centralised servers is subject to deletion, modification, or loss when hosting services terminate. On-chain metadata that references a centralised URI provides no permanence guarantee beyond the lifetime of the hosting provider.

2.5 Partial Execution States

Multi-transaction pipelines produce partial execution states when any step fails or is deliberately omitted. A token may be minted without authority revocation, or a liquidity pool created without locking, if the deployer or platform halts execution after a profitable subset of operations has completed.

3 Prior Art and Existing Mitigations

3.1 Policy-Based Controls

Several platforms offer configurable safety features including optional LP locking, vested buy periods, sell limits, and bonding curve publication. These controls are insufficient because they are administered by the platform operator. A configurable lock can be configured not to lock. A sell limit can be set to 100%. The trust model remains custodial.

3.2 Multi-Step Pipelines

The standard approach to token issuance involves a sequence of transactions: mint, metadata creation, authority revocation, pool creation, and LP locking. Each step is individually verifiable but the pipeline as a whole is not atomic. Any step may be omitted without on-chain evidence of omission.

3.3 Bonding Curve Models

Platforms such as Pump.fun use bonding curves to defer liquidity pool creation until a market cap threshold is reached. This model delays but does not eliminate the vulnerability window. Graduation to a DEX introduces the same LP drain vector that affects direct-launch platforms.

4 The OneForge Protocol

4.1 Core Principle

OneForge replaces policy-based safety guarantees with a two-phase pipeline. Phase one encodes the critical token invariants into a single Solana `VersionedTransaction`: mint creation, supply distribution, authority revocations, and permanent metadata. These operations are indivisible — they succeed together or not at all. Phase two creates the Raydium liquidity pool in a subsequent transaction, guaranteed to execute by an idempotent checkpoint system. The combination eliminates every exploitable window that exists in conventional multi-step pipelines.

4.2 Atomic Execution Model

A Solana transaction is atomic by definition: all instructions within it either succeed together or fail together, with no partial state committed to the ledger. OneForge exploits this property to make the following six operations indivisible:

1. **Token mint creation** — SPL Token mint account initialised with the deployer's specified parameters [6].
2. **Supply minting** — Total supply minted to the platform staging account in a single instruction.
3. **2% supply burn** — Two percent of total supply burned via the SPL Token burn instruction within the same transaction.
4. **On-chain metadata creation** — Token metadata created via the Metaplex Token Metadata program [2], referencing a permanent Arweave URI.
5. **Mint authority revocation** — Mint authority set to null via `setAuthority` [6]. No further supply can ever be issued.
6. **Freeze authority exclusion** — Freeze authority is initialised as `null` in the `initializeMint2` instruction [6]. It is never granted to any party at any point. This is a stronger guarantee than revocation: the authority never existed and cannot be reinstated.

Because these operations are encoded in a single transaction, the following guarantee holds: if the token exists on-chain with mint authority revoked, all six operations have completed successfully.

4.3 Liquidity Pool Seeding

Raydium CPMM pool creation [3] is executed as a second, separate transaction immediately following the atomic token transaction. This is an architectural necessity: the Raydium pool creation instruction set exceeds what can be combined with the token mint instructions within Solana's 1232-byte transaction size limit.

The LP seeding stage is guaranteed by an idempotent checkpoint system. The pipeline records each stage's output to a persistent store before proceeding. If the LP transaction fails or the process is interrupted, it can be safely resumed from the checkpoint without re-executing the atomic token transaction. The pool ID is a deterministic PDA — if the pool already exists on-chain, the stage is skipped. This design means LP seeding either completes or can always be retried; it cannot be silently omitted.

4.4 Transaction Format

The atomic token transaction uses Solana's `VersionedTransaction` v0 format [1]. The combined SPL Token and Metaplex Token Metadata instructions fit within the 1232-byte transaction size limit. The transaction is constructed server-side and submitted with `confirmed` commitment. The LP seeding transaction is constructed separately by the Raydium SDK, also in v0 format, and submitted immediately after confirmation of the token transaction.

4.5 LP Distribution by Tier

OneForge defines two execution tiers with distinct LP distribution rules, both enforced atomically:

Table 1: LP distribution by tier. Platform retains 20% (Quick) or 10% (Safe) of LP tokens.

Tier	Platform fee	LP Burned	LP Locked	Creator LP
Quick	0 SOL	70%	0%	10%
Safe	0.50 SOL	0%	75% (permanent)	15%

Both tiers enforce a minimum LP seed of 0.65 SOL, of which 0.15 SOL is consumed by Raydium's pool creation fee, leaving 0.50 SOL as effective pool liquidity.

5 Permanent State via Arweave

On-chain metadata accounts reference a URI containing the token's name, symbol, description, and image. If this URI points to a centralised server, the metadata is not permanent regardless of the on-chain record's immutability. IPFS pinning depends on active node participation and can be unpinned. Filecoin storage deals expire. Centralised CDNs are subject to service termination, account suspension, and content modification. None of these models provide a URI whose content is guaranteed to resolve identically and permanently from the moment of upload.

5.1 Why Arweave is Architecturally Required

Arweave [4] is not a convenience choice in OneForge — it is a structural requirement. The OneForge atomicity invariant depends on embedding the metadata URI inside the atomic token transaction before that transaction is submitted. This means the URI must exist, and must be permanent, at transaction construction time. A mutable or expirable URI would undermine the atomicity guarantee: the token's on-chain metadata could reference content that later changes or disappears, even though the transaction itself is immutable.

Arweave's permaweb satisfies this requirement uniquely. Content is addressed by the hash of the uploaded data, stored across a decentralised network of miners incentivised by the protocol's endowment model, and retrievable permanently with no recurring cost after the initial upload fee. Once an Arweave transaction is confirmed, the content it references cannot be modified, deleted, or taken down by any party — including the uploader.

Because the URI is included in the transaction that also revokes mint and freeze authority, the following guarantee holds: if a token's authorities are revoked, its metadata is permanently stored on Arweave. The two facts are inseparable.

5.2 Upload Ordering and Transaction Construction

OneForge enforces a strict ordering: Arweave upload completes and the resulting URI is confirmed before the atomic transaction is constructed. The URI is then embedded directly in the Metaplex Token Metadata instruction within the transaction. This ordering is not optional — the Token Metadata program requires the URI at instruction construction time, and the mint authority must still be present to sign the metadata creation instruction before being revoked in the same transaction.

The consequence is that a OneForge token's metadata permanence is established before the token exists on-chain. By the time the mint is visible to any wallet or explorer, the metadata it references has already been written to Arweave permanently.

5.3 Irys and Turbo

Uploads are performed via Irys [5] (formerly Bundlr), a layer-2 bundling service that aggregates Arweave uploads into bundles and provides finality guarantees independent of Arweave's base-layer confirmation time. Irys transactions are

immediately retrievable via the Irys gateway while Arweave base-layer confirmation completes in the background — a property that fits the OneForge pipeline's requirement for low-latency URI availability at transaction construction time.

OneForge is migrating to Turbo, Irys's high-throughput upload service, for reduced upload latency and improved reliability under concurrent launch load. The underlying permanence guarantee is unchanged: all data uploaded via Turbo is stored on Arweave and subject to the same content-addressing and endowment model.

6 Implementation: Crxcible

OneForge is a protocol. Crxcible (crxcible.io) is its reference implementation — a token launchpad live on Solana mainnet that has processed four production launches across both tiers since March 2026 (two Safe, two Quick). The distinction matters: OneForge defines the atomicity invariant and the execution model; Crxcible is one instantiation of that model. Any platform can implement the OneForge protocol.

6.1 Architecture

The Crxcible backend is a Node.js/Express application using BullMQ for asynchronous job processing. The launch pipeline is executed by a dedicated worker with `concurrency: 1` to prevent nonce conflicts on the platform signing wallet. The worker implements a checkpoint pattern using a PostgreSQL database: each stage writes its output before proceeding, enabling safe idempotent retries without duplicating on-chain work.

Phase 1 is signed entirely by the platform wallet and the ephemeral mint keypair. No creator signature is required or requested for the atomic token transaction. The creator's involvement is limited to the payment transaction, which pre-funds the platform wallet with the LP seed SOL before the pipeline begins. This design eliminates any blockhash race condition that would exist if the creator were required to sign: the platform constructs and signs Phase 1 unilaterally, then submits immediately. The worker runs with `concurrency: 1`, ensuring the platform wallet never has two in-flight transactions simultaneously, which eliminates nonce conflicts without requiring durable nonces.

6.2 Production Evidence

The following on-chain transactions demonstrate the OneForge protocol in production:

- Safe tier launch: `4mR2s7N...`
- Safe tier launch: `FCBztQk...`
- Quick tier launch: `9tJx14i...`
- Quick tier launch: `B3mbCcR...`

Each token page displays the mint authority status, freeze authority status, LP lock or burn transaction, and Arweave metadata URI, all derivable from on-chain state without trusting Crxcible's backend.

6.3 Production Incident: Stage 3 Recovery

During production operation, a Quick tier launch failed at Stage 3 (Raydium CPMM pool creation) due to a validation gap that permitted an LP seed below the platform minimum. After Raydium's 0.15 SOL pool creation fee was deducted from the submitted seed, the remaining pool liquidity was insufficient and rejected by the Raydium program. The pipeline's automatic retry logic could not resolve the failure — the error was deterministic, not transient.

The checkpoint system prevented any unsafe state from arising. Stage 2 had already confirmed on-chain: the token was minted, authorities were revoked, and metadata was permanently stored on Arweave. Because Stage 2's output was checkpointed in the database before Stage 3 was attempted, no re-execution of the atomic token transaction was possible or necessary. The partially complete launch was not a rug-pull vector — the token existed in its final, immutable state from the moment Stage 2 confirmed.

Recovery was performed by a purpose-built script that read the existing Stage 2 checkpoint, constructed the pool creation transaction with a corrected LP seed amount, and submitted it directly. The launch completed successfully. No on-chain state was duplicated, no authority was re-granted, and the creator received their launch in full.

This incident demonstrates two properties of the OneForge pipeline under real failure conditions: first, that the atomicity of Stage 2 holds regardless of what happens in subsequent stages — a failed Stage 3 cannot retroactively compromise the token's immutability guarantees; second, that the checkpoint layer is robust

enough to support manual recovery without risk of double-execution, even when automatic retries cannot resolve the underlying error.

6.4 Non-Custodial Design

Crxcible is non-custodial with respect to token authority. The platform does not hold or control any token mint or freeze authority at any point. The creator's LP seed SOL is collected in a payment transaction before the pipeline begins and is used to fund the Raydium pool in Phase 2. The platform signing wallet covers Raydium's pool creation fee and transaction fees, recovering these costs from its LP share.

7 Security Analysis

7.1 Eliminated Attack Vectors

- **Mint authority exploitation**: Mint authority is set to null within the same transaction that creates the mint. There is no post-mint window.
- **Freeze authority exploitation**: Freeze authority is initialised as `null` and is never granted. No wallet can be frozen at any point, including immediately after mint creation.
- **Liquidity drain**: Pool seeding and LP distribution occur atomically. There is no window in which LP tokens are held by the deployer before locking or burning.
- **Metadata impermanence**: Metadata is on Arweave before the transaction is submitted. The URI is immutable once embedded in the on-chain metadata account.
- **Partial execution**: The transaction either succeeds completely or reverts completely. There is no on-chain state representing a partially launched token.

7.2 Residual Risk

OneForge does not protect against market risk, token price manipulation via trading activity, or creator misrepresentation of a token's purpose. It provides structural guarantees about the issuance process, not about the economic behaviour of a token after launch. Buyers remain responsible for evaluating the project behind a token.

8 Future Work

8.1 Token-2022 Extension Support

The Token-2022 program introduces extensions including transfer fees, interest-bearing tokens, and permanent delegate. OneForge's atomic model is compatible with Token-2022 and extension support is planned, enabling the same guarantees for a broader class of token configurations.

8.2 NFT Issuance

NFT issuance via Metaplex Core [2] is the planned next phase of the OneForge protocol. The same atomic guarantees — permanent Arweave metadata, authority exclusion from genesis, and a single-transaction issuance pipeline — apply directly to NFT launches without architectural change.

8.3 Protocol Formalisation

In its current form, OneForge is a specified application-level pipeline: a defined sequence of instructions, construction rules, and invariants that any compliant implementation must satisfy. The guarantees are enforced by Solana's transaction atomicity, not by an on-chain program. Future work includes formalising OneForge as a published on-chain program that other platforms can invoke directly, elevating the protocol's guarantees from implementation-level to smart-contract-level verifiability.

8.4 Native Liquidity Provision

A planned extension of the OneForge protocol is a native liquidity provision service — a lightweight alternative to third-party AMMs, built on an inventory-based pricing model. Rather than a bonding curve, price is derived directly from the ratio of reserves:

$$p = \frac{R_{\text{SOL}}}{R_{\text{token}}}$$

where the spread self-adjusts as inventory is depleted or replenished. This eliminates the graduation event inherent in bonding curve models and the external dependency on a third-party pool program.

The atomic issuance guarantees of OneForge apply in full: pool seeding, reserve

initialisation, and authority revocations are encoded in a single transaction. The result is a trustless liquidity layer that inherits the same structural guarantees as the token launch itself — verifiable on-chain from the first block, with no intermediate state in which reserves can be manipulated.

8.5 Protocol vs. Implementation

It is worth stating explicitly: OneForge is a protocol specification, not a product. Crxcible is a product that implements the OneForge protocol. The guarantees described in this paper derive from the protocol — from Solana's transaction atomicity and the construction rules defined here — not from Crxcible's specific codebase. A reimplementation of OneForge on a different backend, with a different frontend, or under a different brand name would carry the same guarantees, provided it satisfies the same invariants.

8.6 Open Standard

OneForge's atomic issuance model could be adopted as a community standard for trustless token launches on Solana. Publication of this specification is intended to invite review, critique, and adoption by other builders in the ecosystem.

8.7 Decentralised Protocol Execution via AO

The long-term architecture of OneForge is a fully decentralised one. In its current form, the orchestration layer — the pipeline that sequences metadata upload, transaction construction, and post-launch LP distribution — runs on centralised infrastructure. This is an operational dependency, not a protocol requirement. Eliminating it is the next architectural frontier.

AO [7] is a hyper-parallel, decentralised compute layer built on Arweave. Processes run permanently, in parallel, and without a trusted operator. Deploying the OneForge orchestration layer as an AO process would make the protocol genuinely unstoppable: any party could submit a compliant launch, the pipeline would execute without depending on Crxcible's servers, and every state transition would be stored permanently on Arweave as an immutable audit log.

The data layer is already in place. Every OneForge launch writes its metadata to Arweave as the first step of the pipeline — before the Solana transaction is constructed. AO provides the compute layer that closes the loop, combining permanent storage with permanent, verifiable execution.

The vision extends beyond token issuance. A permanent compute layer with access to Arweave's immutable data store is a foundation for a broader class of trustless, verifiable services: on-chain compliance auditing, creator reputation systems, protocol governance, and cross-chain attestation. OneForge's token launch pipeline is the first production-grade service to prove the model. It will not be the last.

9 Conclusion

OneForge demonstrates that the structural vulnerabilities inherent in multi-transaction token issuance pipelines can be eliminated without sacrificing flexibility or user experience. By encoding the full issuance pipeline into a single atomic transaction and combining it with permanent Arweave metadata storage, OneForge provides guarantees that are mathematically enforceable and independently verifiable.

The trust model of existing token launchpads asks buyers to trust that a deployer will complete every post-launch step as promised. OneForge makes that trust unnecessary: every guarantee is mathematically enforced and independently verifiable by anyone, without relying on any claim made by the platform or the deployer.

Crxcible is the first production deployment of the OneForge protocol. The transactions referenced in this paper are live on Solana mainnet and can be verified by anyone without relying on any claim made here.

References

[1] Solana Labs. *Solana Transaction Format v0 and Address Lookup Tables*. docs.solana.com.

[2] Metaplex Foundation. *Token Metadata Standard*. developers.metaplex.com.

[3] Raydium Protocol. *CPMM Pool Creation Specification*. raydium.io.

[4] Arweave. *The Permaweb: Permanent, Decentralised Storage*. arweave.org.

[5] Irys (formerly Bundlr). *Layer-2 Arweave Bundling*. irys.xyz.

[6] SPL Token Program. *Token Program Documentation*. spl.solana.com/token.

[7] Arweave. *AO: Hyper-Parallel Decentralised Computer*. ao.arweave.net.