Figure Q3 2025: Hastra vault-mint & vault-stake Solana programs

Last updated: November 25th, 2025
Authors: Carlos Rodriguez, Aleksandar Stojanovic

Introduction

In September 2025, Figure engaged Informal Systems to conduct a comprehensive security audit of the Hastra `vault-mint` and `vault-stake` Solana programs. This document provides a high-level summary of our findings and recommendations.

Engagement overview

The audit was conducted in two phases between September 24th and November 18th, 2025, by Carlos Rodriguez and Aleksandar Stojanovic.

Initial audit (September 24th - October 3rd, 2025)

The engagement began with the programs residing in separate repositories:

- `hastra-sol-vault-mint` at commit `b881f84`.
- `hastra-sol-vault-stake` at commit `4bbc0b3`.

Our methodology included threat modeling and manual code review, focusing on security properties, access control mechanisms, and integration correctness.

Follow-up Review (November 2025)

After the development team addressed our initial feedback and consolidated both programs into a unified repository (`hastra-sol-vault`), we conducted a second review of the updated implementation at commit `0b55b4a`. This review evaluated the fixes and examined the new

architectural changes, including cross-program invocation (CPI) mechanisms between the two programs.

System overview

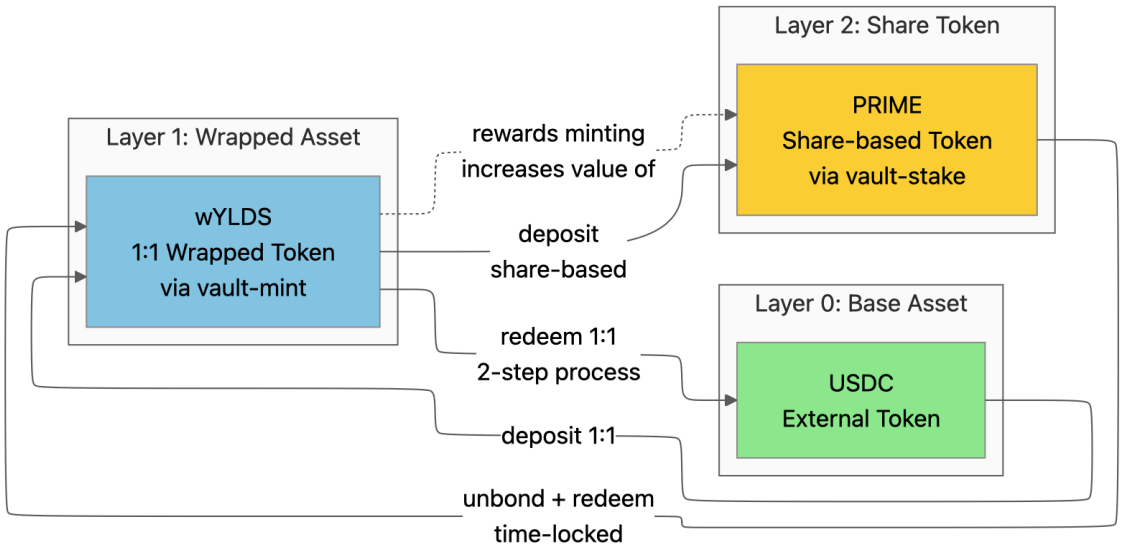
The Hastra protocol consists of two interconnected Solana programs that create a layered staking and liquidity system.

Architecture

The `vault-mint` program serves as the base layer, operating as a custodial wrapper that accepts USDC deposits and issues wYLDS (wrapped YLDS) tokens at a 1:1 ratio. Users can redeem their wYLDS tokens back to USDC through an admin-mediated two-step process. The program includes merkle-tree-based reward distributions where users claim allocated rewards by providing cryptographic proofs.

The `vault-stake` builds on this foundation by accepting wYLDS deposits and issuing PRIME share tokens based on a dynamic exchange rate calculated from the vault's wYLDS balance and PRIME token supply. Unlike `vault-mint`, this program implements an unbonding period before redemptions. Users must first request to unbond their position, wait for the configured unbonding period to elapse, and then complete their redemption.

A key interaction occurs through the rewards mechanism: `vault-stake`'s `publish_rewards` function performs a CPI to `vault-mint`'s `external_program_mint`, minting new wYLDS tokens directly into `vault-stake`'s vault, thereby increasing the wYLDS-per-PRIME ratio and distributing yield to stakers.



Both programs include account freeze/thaw functionality for their respective tokens and maintain administrator lists for governance.

Audit findings and conclusions

Initial audit results

Our initial assessment identified 9 findings across various severity levels: 1 Critical, 4 Medium, 2 Low, and 2 Informational.

The critical vulnerability involved missing vault token account ownership validation in the deposit operations of both programs. The programs only verified that the vault token account had the correct mint type but failed to ensure the account was owned by the program-controlled vault authority PDA. This allowed attackers to provide their own vault token accounts, enabling them to redirect user deposits while still receiving mint tokens in exchange—effectively breaking the fundamental backing guarantee of the protocol.

> The issue was resolved by the development team in commit `0b55b4a` of `hastra-sol-vault`. The resolution was reviewed by the audit team.

Medium severity findings included insufficient space allocation for `Config` accounts (failing to account for Borsh serialization `Vec` length prefixes), missing parameter validation for the unbonding period in `vault-stake` (allowing accidental misconfiguration), missing validation that vault and mint tokens are distinct during initialization, and missing program update authority validation in the initialize instruction (creating a race condition where any account could front-run deployment).

The low and informational findings focused on code quality improvements: inconsistent bump seed usage, missing rewards administrators length validation, lack of comprehensive logging for state-changing operations, and various opportunities to leverage Anchor's modern features more effectively.

Follow-up review findings

During our November review of the consolidated repository, we identified two additional issues that required attention.

First, we discovered a vault inflation attack vulnerability in the `vault-stake` program's deposit function. The share calculation read the vault balance before the user's deposit was transferred, making it vulnerable to manipulation through direct token donations. An attacker could make a minimal first deposit, donate a large amount of tokens directly to the vault to inflate the share price, and then profit from subsequent

victims' deposits that would round down to zero shares due to integer division. We recommended implementing the OpenZeppelin ERC4626 mitigation strategy using virtual shares and asset offsets.

> The issue was resolved by the development team in commit `2c792be` of `hastra-sol-vault`. The resolution was reviewed by the audit team.

Second, we found insufficient CPI origin verification for the external mint operation. The `vault-mint` program's `external_program_mint` instruction didn't restrict minting to CPIs from the `vault-stake` program. While it checked that an executable account matching the allowed program existed, it didn't verify that the current instruction was actually being executed by the `vault-stake` program. Any rewards administrator could call the function directly. We recommended introducing a PDA-based authentication mechanism where only the `vault-stake` program could sign for a specific PDA during the CPI call.

> The issue was resolved by the development team in commit `2c792be` of `hastra-sol-vault`. The resolution was reviewed by the audit team.

Development team recognition

The Hastra programs demonstrate solid engineering practices with well-structured code, comprehensive use of Anchor's constraint system, proper PDA derivation patterns, and robust role-based access control. The architecture effectively leverages modern Solana best practices and shows thoughtful consideration of security principles throughout the design.

We commend the Figure development team for their exceptional responsiveness and collaboration throughout the audit process. They promptly addressed all identified vulnerabilities, implementing our recommendations with careful attention to detail. The team engaged constructively with our feedback, asked clarifying questions when needed, and demonstrated a strong commitment to security best practices. All findings were thoroughly resolved, with the team going beyond mere fixes to improve overall code quality and maintainability.

The consolidation of both programs into a unified repository, combined with the implementation of our recommendations, has resulted in a robust and secure foundation for the Hastra ecosystem. The comprehensive and well-structured test suite—covering all core functionalities of the `vault-mint` and `vault-stake` programs, along with extensive security and edge-case scenarios—further strengthens confidence in the system's correctness. With the identified issues resolved, the rigorous testing in place, and the enhanced security measures implemented, these programs are well-positioned for production deployment.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability, etc.) set forth in the associated Services Agreement. This report provided in connection with the Services set forth in the Services Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement.

This audit report is provided on an “as is” basis, with no guarantee of the completeness, accuracy, timeliness or of the results obtained by use of the information provided. Informal has relied upon information and data provided by the client, and is not responsible for any errors or omissions in such information and data or results obtained from the use of that information or conclusions in this report. Informal makes no warranty of any kind, express or implied, regarding the accuracy, adequacy, validity, reliability, availability or completeness of this report. This report should not be considered or utilized as a complete assessment of the overall utility, security or bugfree status of the code.

This audit report contains confidential information and is only intended for use by the client. Reuse or republication of the audit report other than as authorized by the client is prohibited.

This report is not, nor should it be considered, an “endorsement”, “approval” or “disapproval” of any particular project or team. This report is not, nor should it be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts with Informal to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor does it provide any indication of the client’s business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should it be leveraged as investment advice of any sort.

Blockchain technology and cryptographic assets in general and by definition present a high level of ongoing risk. Client is responsible for its own due diligence and continuing security in this regard.