



# SECURITY ASSESSMENT

Provided by Accretion Labs Pte Ltd. for The Vault  
January 23, 2025  
A24VAU1



## AUDITORS

Role	Name
Lead Auditor	Robert Reith (robert@accretion.xyz)

## CLIENT

**The Vault** (<https://thevault.finance/>) The Vault is a staking pool on the Solana blockchain which aggregates SOL to stake with community validators through a liquid staking token vSOL.

## ENGAGEMENT SCOPE

### The Vault finance directed stake program

**Link:** <https://github.com/SolanaVault/directed-stake/tree/master/program/s/directed-stake>

**Commit:** e940a48e94ad452c064b13eb94a2e0fb8b335ea9

**ProgramID:** DStkUE3DjxBhVwEGNzv89eni1p7LpYuHSxxm1foggbEv

### The Vault finance directed stake token program

**Link:** <https://github.com/SolanaVault/directed-stake/tree/master/program/s/directed-stake-token>

**Commit:** e940a48e94ad452c064b13eb94a2e0fb8b335ea9

**ProgramID:** VtokuDkNTQxXPJAp37ZXsoS4myJpxjHZ5RmCbDhUED4

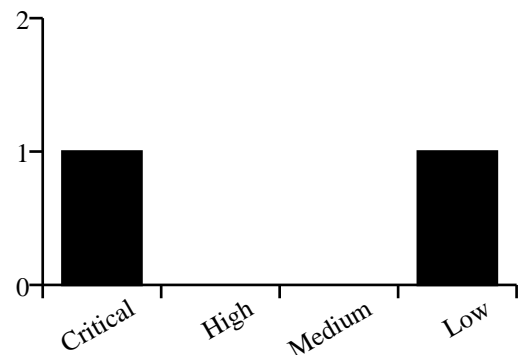
## ENGAGEMENT TIMELINE



## ASSESSMENT

The Directed-Stake Token (DST) programs within The Vault enable DST creators to designate specific validators for stake allocation. This mechanism ensures all staked assets are automatically delegated to predetermined validator addresses. Our security analysis revealed a robust architectural design, with a single critical vulnerability stemming from a copy-paste error that was promptly remediated. Several low-severity optimizations were recommended and implemented during the evaluation period. The swift resolution of identified issues demonstrates strong operational security practices in the development lifecycle.

## SEVERITY DISTRIBUTION



## VERIFIED ON-CHAIN CODE

### Program 1

**ProgramID:** DStkUE3DjxBhVwEGNzv89eni1p7LpYuHSxxm1foggbEv  
**Repository:** <https://github.com/SolanaVault/directed-stake/>  
**Build Hash:** d9ff7a00d6548f12c204db2705728431385a63f8b8c63ec70f4e61131ee864fc  
**Commit:** 74417126222dc3dd472a8572f10bb3cf9ecdbf73

### Program 2

**ProgramID:** VtokuDkNTQxXPJAp37ZXsoS4myJpxjHZ5RmCbDhUED4  
**Repository:** <https://github.com/SolanaVault/directed-stake/>  
**Build Hash:** c68ebdbb303e2b8183e90f2fb0c24f291a83824223c09652609b36ad44f42ffd  
**Commit:** 74417126222dc3dd472a8572f10bb3cf9ecdbf73

# ISSUES SUMMARY

ID	TITLE	SEVERITY	STATUS
ACC-C1	BurnDST is Missing Constraints Which Leads to LOF	critical	fixed
ACC-L1	Third party can DoS CreateDST token account creation	low	fixed

## DETAILED ISSUES

ID	ACC-C1
Title	BurnDST is Missing Constraints Which Leads to LOF
Severity	critical
Status	fixed

### Description

We found that the `BurnDST` instruction does not have any constraints, which means that it can be called almost without any restrictions on the passed accounts. An attacker could create a random Mint and issue some tokens, and then burn them using ``BurnDST`` to receive ``vSol``, effectively draining the pool by burning worthless tokens.

### Location

<https://github.com/SolanaVault/directed-stake/blob/e940a48e94ad452c064b13eb94a2e0fb8b335ea9/programs/directed-stake-token/src/lib.rs#L461C1-L483C2>

### Relevant Code

```
#[derive(Accounts)]
pub struct BurnDST<'info> {
    /// The DST account
    #[account(mut)]
    pub dst: AccountLoader<'info, DSTInfo>,
    /// Mint account of the DST.
    #[account(mut)]
    pub token_mint: Account<'info, Mint>,
    /// The DST token account to burn from
    #[account(mut)]
    pub source_dst_account: Account<'info, TokenAccount>,
    /// The owner of the source_dst_account to burn from
    #[account(mut)]
    pub source_authority: Signer<'info>,
    /// The account to send vSOL to
    #[account(mut)]
    pub destination_vsol_account: Account<'info, TokenAccount>,
    /// The vSOL reserves token account owned by the DST.
    #[account(mut)]
    pub dst_vsol_reserves: Account<'info, TokenAccount>,
    /// SPL token program
    pub token_program: Program<'info, Token>,
}
```

### Mitigation Suggestion

We recommend adding the same constraints as for `MintDST`.

### Remediation

The issue was fixed in commit `61cffe37124ba2b6af6553e6cd345670260fb249`.

ID	ACC-L1
Title	Third party can DoS CreateDST token account creation
Severity	low
Status	fixed

## Description

We found that the `CreateDST` instruction uses ``init`` to create an associated token account. However, because it is an ATA, anyone could create this token account before this instruction is called, which would result in the instruction failing.

## Location

<https://github.com/SolanaVault/directed-stake/blob/e940a48e94ad452c064b13eb94a2e0fb8b335ea9/programs/directed-stake-token/src/lib.rs#L281C1-L319C2>

## Relevant Code

```
#[derive(Accounts)]
pub struct CreateDST<'info> {
    /// [...]
    /// vSOL mint.
    #[account(address = VSOL_ADDRESS)]
    pub vsol_mint: Account<'info, Mint>,
    #[account(
        init,
        payer = payer,
        associated_token::mint = vsol_mint,
        associated_token::authority = dst,
    )]
    pub vsol_reserves: Account<'info, TokenAccount>,

    /// [...]
    pub associated_token_program: Program<'info, AssociatedToken>,
}
```

## Mitigation Suggestion

Use `init_if_needed` instead of ``init`` when using ATAs.

## Remediation

This issue has been fixed in commit `61cffe37124ba2b6af6553e6cd345670260fb249`

## APPENDIX

### Vulnerability Classification

We rate our issues according to the following scale. Informational issues are reported informally to the developers and are not included in this report.

Severity	Description
<b>Critical</b>	Vulnerabilities that can be easily exploited and result in loss of user funds, or directly violate the protocol's integrity. Immediate action is required.
<b>High</b>	Vulnerabilities that can lead to loss of user funds under non-trivial preconditions, loss of fees, or permanent denial of service that requires a program upgrade. These issues require attention and should be resolved in the short term.
<b>Medium</b>	Vulnerabilities that may be more difficult to exploit but could still lead to some compromise of the system's functionality. For example, partial denial of service attacks, or such attacks that do not require a program upgrade to resolve, but may require manual intervention. These issues should be addressed as part of the normal development cycle.
<b>Low</b>	Vulnerabilities that have a minimal impact on the system's operations and can be fixed over time. These issues may include inconsistencies in state, or require such high capital investments that they are not exploitable profitably.
<b>Informational</b>	Findings that do not pose an immediate risk but could affect the system's efficiency, maintainability, or best practices.

### Audit Methodology

Accretion is a boutique security auditor specializing in Solana's ecosystem. We employ a customized approach for each client, strategically allocating our resources to maximize code review effectiveness. Our auditors dedicate substantial time to developing a comprehensive understanding of each program under review, examining design decisions, expected and edge-case behaviors, invariants, optimizations, and data structures, while meticulously verifying mathematical correctness—all within the context of the developers' intentions.

Our audit scope extends beyond on-chain components to include associated infrastructure, such as user interfaces and supporting systems. Every audit encompasses both a holistic protocol design review and detailed line-by-line code analysis.

During our assessment, we focus on identifying:

- Solana-specific vulnerabilities
- Access control issues
- Arithmetic errors and precision loss
- Race conditions and MEV opportunities
- Logic errors and edge cases
- Performance optimization opportunities
- Invariant violations
- Account confusion vulnerabilities
- Authority check omissions
- Token22 implementation risks and SPL-related pitfalls
- Deviations from best practices

Our approach transcends conventional vulnerability classifications. We continuously conduct ecosystem-wide security research to identify and mitigate emerging threat vectors, ensuring our audits remain at the forefront of Solana security practices.