PRISMSEC
End-To-End Blockchain
Security Solutions
Auditing, Penetration Testing,
Adversary Simulation, AI Security Testing, & More

Tharwa sThUSD Security Review

By Prism

X

Auditors

Lead Security Researcher: Jacobo Lansac (@Jacopod)

August 24, 2025

Contents

1 Introduction

A time-boxed review of **Tharwa sThUSD** smart contracts, developed by Tharwa. The focus of this review is to identify security vulnerabilities, explain their root-cause and provide solutions to mitigate the risk.

Gas optimizations are not the main focus but will also be identified if found.

2 About Prism

Prism delivers specialized security solutions for blockchain and AI companies. We go beyond traditional audits, offering bespoke penetration testing, adversary simulation, and AI security solutions to meet the needs of every client. With tailored services and best-in-class expertise, we safeguard your business against the most sophisticated threats, allowing you to focus on innovation.

Learn more about us at prismsec.xyz

3 About Tharwa

Tharwa Finance aims the first RWA-Collateralized Stablecoin backed by an AI-Driven RWA hedge fund. They aim to do so by tokenizing diversified multi-asset funds to power RWA-backed stablecoin yields.

4 Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time and resource-bound effort to find as many vulnerabilities as possible, but there is no guarantee that all issues will be found. This security review does not guarantee against a hack. Any modifications to the code will require a new security review.

This review does not focus on the correctness of the happy paths. Instead, it aims to identify potential security vulnerabilities and attack vectors derived from an unexpected and harmful usage of the contracts. The devs are ultimately responsible for the correctness of the code and its intended functionality.

A security review is not an endorsement of the underlying business or product and can never be taken as a guarantee that the protocol is bug-free.

5 Risk classification

Severity level	Impact: High	Impact: Medium	Impact: Low
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

5.1 Impact

- **High**: leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- **Medium**: only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- **Low**: can lead to unexpected behavior with some of the protocol's functionalities that are not so critical.

5.2 Likelihood

- **High**: almost certain to happen, easy to perform or highly incentivized.
- **Medium**: only conditionally possible or incentivized, but still relatively likely
- **Low**: requires multiple unlikely conditions, or little-to-no incentive

5.3 Action required for severity levels

- **Critical**: Must fix as soon as possible (if already deployed)
- **High**: Must fix (before deployment if not already deployed)
- **Medium**: Should fix
- **Low**: Could fix

6 Executive Summary

Scope details

Project Name	Tharwa
Type of Project	Staking, ERC4626 vault, Yield
Repository	tharwa-finance/sthUSD
Review commit	666ac75d904fcff2817cda16935e7f8154f2b301
Mitigation commit	29057c866ab92c55615e0258cda80dd904882540
Audit Timeline	2025-08-22 - 2025-08-24
Methods	Manual Review, Testing

Files in scope	**nSLOC**
src/sThUSD.sol	232
src/ThUSDSilo.sol	21
Total	**253**

Issues Found

Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	1

7 Architecture review

7.1 Architecture review

sThUSD is an ERC-4626 vault that allows users to stake thUSD tokens and earn protocol yield that vests linearly over time. It includes donation-attack protection, optional fees, and a cooldown mechanism for exits, that can be toggled by admins.

7.1.1 Entry Points

All classic deposits of an ERC4626 (deposit, mint, withdraw, redeem), where the last two are only enalbed if cooldown is disabled. If cooldown is enabled, `redeem()` and `withdraw()` are substituted by `cooldownShares()` and `cooldownAssets()` respectively, which have the same effect with the difference that funds can only be actually received by the user at the end of the cooldown.

After the cooldown, fund are retrieved calling `unstake()`.

Admin Functions:

- `addYield(amount)` - Add yield that vests over vestingPeriod
- setters: `setVestingPeriod(period)`, `setCooldownPeriod(period)`, `setFees(...)`

7.1.2 Main Mechanics

1. Share Pricing: totalAssets() / totalSupply() where totalAssets() excludes unvested yield
2. Yield Vesting: New yield vests linearly over 30 days (configurable), protecting against donation attacks
3. Cooldown System: When enabled (7 days default), users must cooldown → wait → unstake instead of direct withdrawals
4. Fees: Optional entry/exit fees (max 10% each) sent to treasury

7.1.3 Architecture assessment

- The contracts are written with security in mind, favoring simplicity over complexity.
- The architecture is very clean and well strcutured.
- The test suite is very complete.

8 Findings

8.1 Low Risk

8.1.1 [L1] - Step-wise jumps in share pricing if the period duration is updated during an active vesting

Location:

Contract: `sThUSD.sol`, Function: `setVestingPeriod()`

Description:

The `setVestingPeriod()` function allows updating the vesting period even when a vesting period is currently active. This causes a step change in the share price because the `_unvestedAmount()` calculation immediately reflects the new period duration, changing how much yield is considered "vested" versus "unvested" for share pricing purposes.

Impact:

When the vesting period is changed during active vesting:

- If the new period is shorter: A higher percentage of funds become vested, causing a step-wise increase in share price.
- If the new period is longer: Funds that were already vested become unvested, causing a step-wise decrease in share price.

This sudden price change can be exploited through sandwich attacks - buying shares before the change and selling after to profit from the price step. These attacks would only be profitable if the step is greater than the entry/exit fees together.

Severity:

- Probability: medium, as vesting periods are long and are expected to constantly generate yield.
- Impact: med as it impacts users' yield.

Mitigation:

Only allow changing the vesting duration when no vesting is active, similar to how `addYield()` already prevents adding yield during active vesting.

Code Context:

```
function setVestingPeriod(uint256 newPeriod) external onlyRole(DEFAULT_ADMIN_ROLE) {
    if (newPeriod == 0) revert PeriodZero();
+   if (_unvestedAmount() != 0) revert VestingActive();
    vestingPeriod = newPeriod;
    emit VestingPeriodSet(newPeriod);
}
```

Response: Fixed

Fixed in 29057c866ab92c55615e0258cda80dd904882540

8.2 Informationals

Location

sThUSD.sol, in functions `cooldownAssets()` and `cooldownShares()`.

Description

If `cooldownPeriod == 0`, it reverts saying `CooldownActive()`, but the cooldown is precisely inactive when `cooldownPeriod == 0`, so the error message is confusing and should be the opposite.

Suggestoin

```
    + error CooldownNotActive();

    function cooldownAssets(
        uint256 assets
    ) external whenNotPaused returns (uint256 shares) {
-       if (cooldownPeriod == 0) revert CooldownActive(); // operation disabled when cooldown is off
+       if (cooldownPeriod == 0) revert CooldownNotActive(); // operation disabled when cooldown is off
        // ...
    }
```