Smart Contract Security Audit Report

Auditor: Your Name

January 24, 2026

Contents

High Severity Findings

[H-1] Erroneous `updateExchangeRate` in `deposit` Causes Redemption Failure

Impact: High
Likelihood: High

Description: In the ThunderLoan system, the `exchangeRate` determines the conversion between AssetTokens and underlying tokens. It is responsible for tracking fees owed to liquidity providers.

However, the `deposit` function updates the exchange rate without actually collecting any fees, leading the protocol to believe it holds more assets than it does.

Vulnerable Code:

```
function deposit(IERC20 token, uint256 amount)
    external
    revertIfZero(amount)
    revertIfNotAllowedToken(token)
{
    AssetToken assetToken = s_tokenToAssetToken[token];
    uint256 exchangeRate = assetToken.getExchangeRate();

    uint256 mintAmount =
        (amount * assetToken.EXCHANGE_RATE_PRECISION()) / exchangeRate;

    assetToken.mint(msg.sender, mintAmount);

    // Incorrect exchange rate update
    uint256 calculatedFee = getCalculatedFee(token, amount);
    assetToken.updateExchangeRate(calculatedFee);

    token.safeTransferFrom(msg.sender, address(assetToken), amount);
}
```

Impact:

1. Liquidity providers are unable to redeem funds.
2. Fees are miscalculated, over-rewarding or under-rewarding LPs.

Proof of Concept:

1. Liquidity provider deposits funds.
2. A user executes a flash loan.
3. LP redemption becomes impossible.

Recommended Mitigation: Remove the exchange rate update from the `deposit` function.

—

[H-2] Using `deposit` Instead of `repay` Allows Theft of Funds

Impact: High
Likelihood: High

Description: An attacker can repay a flash loan using `deposit`, satisfying the balance check while minting AssetTokens that can later be redeemed, draining protocol funds.

Attack Flow:

1. Attacker takes a flash loan.
2. Funds are deposited instead of repaid.
3. Flash loan check passes.
4. Attacker redeems minted AssetTokens.

Recommended Mitigation: Prevent deposits while a flash loan is active.

```
if (s_currentlyFlashLoaning[token]) {
    revert ThunderLoan__CurrentlyFlashLoaning();
}
```

—

[H-3] Storage Collision in Upgrade Freezes Protocol

Impact: High
Likelihood: High

Description: The storage layout between `ThunderLoan` and `ThunderLoanUpgraded` is inconsistent, causing storage collisions due to reordered variables and constants.

Impact: Fee calculations become corrupted, breaking core protocol functionality.

Recommended Mitigation: Maintain identical storage layout across upgrades.

```
uint256 public constant FEE_PRECISION = 1e18;
uint256 private s_flashLoanFee;
```

Medium Severity Findings

[M-1] TSwap Oracle Manipulation via Flash Loans

Impact: Medium
Likelihood: Medium

Description: ThunderLoan relies on TSwap AMM pricing, which is vulnerable to manipulation within a single transaction using flash loans.

Impact: Liquidity providers receive reduced fees due to manipulated oracle prices.

Recommended Mitigation: Use Chainlink price feeds with a TWAP fallback.

Disclaimer

This audit does not guarantee the absence of vulnerabilities. It reflects the auditor's best effort at the time of review.