Granite Protocol Audit Report

by: Halipot (Discord: halipot123)

2024.7

CR-01 Share token price manipulation resulted in assets being stolen

Info

Granite Protocol adopts the ERC4626 share token vault model, which has a common security issue known as the share price inflation attack. For more details, please visit https://tienshaoku.medium.com/eip-4626-inflation-sandwich-attack-deep-dive-and-how-to-solve-it-9e3e320cc3f1

Desc

Granite Protocol calculates the number of shares that should be obtained through asset * total-shares / total-asset in the deposit function. The key point of share price inflation lies here. If asset * total-shares < total-asset, due to rounding, 0 shares will be received, causing losses for the liquidity-provider. Under normal circumstances, this is unlikely, but I have discovered a combination of vulnerabilities that can lead to the controllability of the total-asset value, making share price inflation attacks feasible.

Impact

As long as the attacker monitors the on-chain transactions that have not been packaged, and controls the share price in advance, they can steal funds from subsequent depositors.

Code Analysis

The manipulation of total-asset requires a combination of several conditions.
First, let's analyze the borrow function in **borrower.clar.**

There is a judgment: (asserts! (>= (unwrap-panic (contract-call? .mock-usdc get-balance .state)) amount) ERR-INSUFFICIENT-FREE-LIQUIDITY)
The code does not check whether the usdc deposited by the liquidator-provider in state.calr is greater than the borrowed amount. It determines whether the loan can be made by obtaining the balance of usdc through get-balance, which can be bypassed.
By using the transfer function of usdc, usdc can be deposited into the **state.clar** contract, bypassing the previous judgment. In this case, even if the liquator-provider does not provide sufficient usdc, the borrower can still complete the borrowing operation. The consequence is that open-interest is much larger than total-asset.
When calling the accrue-interest function to accumulate interest rates, the utilization rate `(/ (* open-interest-fixed one-12) total-assets-fixed)` needs to be calculated. This utilization rate will be very large, and `interest-rate-per-block` is calculated based on the `utilization rate`, which will also become very large.
Finally, it participates in the calculation and accumulates interest rates, causing `total-asset` to increase while `total-shares` remain unchanged. This leads to a very small total-share/total-asset ratio and a very low share price. Subsequent users will lose a large amount of funds due to the rounding down of the assset * total-shares / total-asset calculation. The attacker only needs to deposit a small amount of usdc in advance to obtain a large amount of funds through withdrawal in subsequent operations.

Reproduction Steps

1. Prepare an attacker user Bob and a victim John in advance.

2. Bob calls the deposit function of the liqudator-provider.clar contract and deposits 1 usdc to obtain 1 share.
3. Bob calls the add-collateral function of the borrower.clar contract to obtain a certain borrowing limit.
4. Bob transfers 1e8 usdc to the state.clar contract through the transfer function of usdc.
5. Bob calls the borrow function of borrower.clar to borrow 0.99e8 usdc.
6. John calls the deposit function of liqudator-provider.clar and deposits a large amount of funds, such as 1800000e8.
7. Bob calls the reedem function of liqudator-provider.clar to steal a large amount of funds.

Recommendation

Fix Use a variable to record the remaining usdc in the state contract. When calculating the utilization rate, edge cases should be considered, and an upper and lower limit should be set for it. For example, the maximum utilization rate is 100%.

Remediation review

Resolved. Add tracking balances in state variables

H-1 Lacks support for collateral tokens with different decimal places

location

- https://github.com/Trust-Machines/granite/blob/da157a320f63c62a09fcaf357e5a670b820a123c/contracts/borrower.clar#L233
- https://github.com/Trust-Machines/granite/blob/da157a320f63c62a09fcaf357e5a670b820a123c/contracts/liquidator.clar#L231

Currently, the contract calculations only consider the case of tokens with 8 decimal places, without taking into account the situation of tokens with other decimal places. This will cause the values obtained by `liquidator.clar` and `borrower.clar`'s `get-collateral-value` function to deviate from the actual values

Recommendation

`https://github.com/Trust-Machines/granite/blob/da157a320f63c62a09fcaf357e5a670b820a123c/contracts/state.clar#L22C41-L23C1`

modify the collaterals map by adding a decimals variable, and then when calculating the price, use the `toFixed` function to convert it to eight decimal places

Remediation review

Resolved. By adding new tuples in the collaterals to record decimals.

M-1 An imperfect if statement may result in collateral-to-give being greater than collateral-value.

location

- https://github.com/Trust-Machines/granite/blob/da157a320f63c62a09fcaf357e5a670b820a123c/contracts/liquidator.clar#L138

The if statement at `https://github.com/Trust-Machines/granite/blob/da157a320f63c62a09fcaf357e5a670b820a123c/contracts/liquidator.clar#L138` is not complete enough. In some edge cases, it may lead to a situation where `collateral-value < repay-amount == total-repay-amount < liquidator-repay-amount`, resulting in `collateral-to-give` being greater than the user's `collateral-value`, which could cause the contract to run in an abnormal state.

Recommendation

Add a check to ensure that `collateral-to-give` cannot be greater than the user's `collateral-value`.

Remediation review

Resolved. Add bad debt liquidate logic.

M-2 In some cases, using tx-sender to determine permissions may lead to phishing attacks.

location

1. https://github.com/Trust-Machines/granite/blob/da157a320f63c62a09fcaf357e5a670b820a123c/contracts/governance.clar#L452
2. https://github.com/Trust-Machines/granite/blob/da157a320f63c62a09fcaf357e5a670b820a123c/contracts/governance.clar#L456
3. https://github.com/Trust-Machines/granite/blob/da157a320f63c62a09fcaf357e5a670b820a123c/contracts/borrower.clar#L29
4. https://github.com/Trust-Machines/granite/blob/da157a320f63c62a09fcaf357e5a670b820a123c/contracts/borrower.clar#L153

In Clarity, `tx-sender` is similar to `tx.origin` in Solidity, both of which are susceptible to phishing attacks. Using `tx-sender` to determine `is-governance-member` is highly prone to phishing attacks. In Solidity, using `msg.sender` instead of `tx.origin` is a safer solution. To ensure security without compromising composability, using `contract-caller` to identify users is a more secure choice.

Recommendation

Replace tx-sender with contract-caller.

Remediation review

Resolved.

M-3 Use the less manipulable EMA price instead of the read price.

location

- https://github.com/Trust-Machines/granite/blob/da157a320f63c62a09fcaf357e5a670b820a123c/contracts/modules/pyth-oracle.clar#L75

When reading oracle price, using the EMA (Exponential Moving Average) price is a safer choice. Tokens with low liquidity are easily susceptible to price manipulation. By using the EMA price, which calculates a weighted average over a period of time, you can avoid the impact of short-term price fluctuations. This is also a common practice in DeFi.

Recommendation

`(price (get ema-price pyth-record))`

Remediation review

Known. It might cause other issues, so we decided not to use ema-price.

M-4 Additional checks in transfers may pose a theft risk to other integrated DeFi contracts.

location

- https://github.com/Trust-Machines/granite/blob/da157a320f63c62a09fcaf357e5a670b820a123c/contracts/state.clar#L438

Common transfer checks usually include `(is-eq tx-sender sender)`. Adding an additional check like `(is-eq contract-caller sender)` may pose a theft risk to other integrated DeFi protocols. If another DeFi protocol calls the `state.clar` transfer function and the `sender` variable is controllable, then if `contract-caller` equals `sender`, an attacker could potentially theft Granite LP Tokens from the integrated DeFi protocol.

Recommendation
Remove the `(is-eq contract-caller sender)` statement in `https://github.com/Trust-Machines/granite/blob/da157a320f63c62a09fcaf357e5a670b820a123c/contracts/state.clar#L438`

Remediation review
Known.

Review after fix

H-1 liquidate calc-collateral-to-give function miss decimals converted

location:

- https://github.com/Trust-Machines/granite/blob/da157a320f63c62a09fcaf357e5a670b820a123c/contracts/liquidator.clar#L194

When calculating calc-collateral-to-give, the decimals are not converted. This can lead to asset loss when handling collateral tokens with decimals other than 8. (collateral-amount (try!

(safe-div (* repay-amount-with-discount resolution-factor) collateral-price))) When the resolution-factor is equal to 1e8, the calculated calc-collateral-to-give is fixed at 8 decimals. If the collateral-token does not have 8 decimals, it can lead to a loss of funds.

Recommendation
The collateral-amount should be equal to repay-amount-with-discount * decimals / collateral-price.

Remediation review
Resolved. (decimal-corrected-collateral (contract-call? .math to-fixed collateral-amount MARKET-TOKEN-DECIMALS collateral-decimals)
Convert based on the decimals of the collateral

M-1 repeated contract-caller validation and contract-caller validation may be result other protocols cannot be integrated.

location:

- https://github.com/Trust-Machines/granite/blob/e554a90c52b4e962a58e9b7a2128a87566014d66/contracts/state.clar#L413

Is-eq contract-caller sender The check is repeated, and itt should be replaced by (is-eq tx-sender sender) here, because assuming that there is a contract that wants to interact with Grantie Lp token and provides a stake function. If the stake is a certain Granite Lp token can obtain income, then the contract call Granite Lp Token's transfer function cannot complete the stake, because the contract-caller is currently checked, and the function can only be realised by tx-sender. In solidity, approve and transferFrom

are generally used to complete, but there is generally no such paradigm in clarity.

Recommendation

only check if tx-sender equals sender

Remediation review

Resolved.