G0 Group

Security Review of

Hats.Finance

October 2022

Hats.Finance / October 2022

Files in scope

All soldity files in

https://github.com/hats-finance/hats-contracts/tree/1272ef58df5d65fcc8d4cddbb23772cd4a6a5cf3/contracts

Current status

All discovered issues have been fixed by the developer. There are no known issues in the relevant contracts in:

https://github.com/hats-finance/hats-contracts/tree/309d0832fd5a778a04c38a0d74c9c7eee3589449/contracts

Issues

1. Check ensuring withdraw request can't be submitted until the current request expires can be bypassed

type: security / severity: medium

This check: HATVault.sol#L498 can be bypassed by doing a zero token transfer, which will always succeed and will reset `withdrawEnableStartTime[from]` to `0` see: HATVault.sol#L725

status - fixed

Issue has been fixed and is no longer present in

https://github.com/hats-finance/hats-contracts/tree/309d0832fd5a778a04c38a0d74c9c7eee3589449/contracts

2. Incorrectly defined time intervals allow users to send and receive shares at the same time

type: security / severity: medium

Intervals defined on following lines overlap:

HATVault.sol#L714

HATVault.sol#L763

This means that there is a possibility a block will have a timestamp that allows user to receive shares and send them in the same block, this is an unexpected behavior.

status - fixed

Issue has been fixed and is no longer present in

https://github.com/hats-finance/hats-contracts/tree/309d0832fd5a778a04c38a0d74c9c7eee3589449/contracts

3. Self-transfer of shares allows minor manipulation of RewardController.rewardDebt variable

type: security / severity: medium

Due to issue #2, users can transfer shares to themselves, this will lead to weird behavior due to the `RewardController.commitUserBalance` calls, most likely ending in an underflow error, but rounding issues might allow the user to bypass the underflow error for very small transfers and manipulate the `RewardController.rewardDebt` variable.

status - fixed

Issue has been fixed and is no longer present in

https://github.com/hats-finance/hats-contracts/tree/309d0832fd5a778a04c38a0d74c9c7eee3589449/contracts

4. User’s ability to transfer shares can be blocked by anybody by 0 amount transferFrom call

type: security / severity: critical

A user can call `HATVault.transferFrom` with `0` amount and arbitrary from which will always pass. This will lead to `withdrawEnableStartTime[from]` being set to `0`. This allows anybody to block all transactions of any address.

status - fixed

Issue has been fixed and is no longer present in

https://github.com/hats-finance/hats-contracts/tree/309d0832fd5a778a04c38a0d74c9c7eee3589449/contracts

5. User’s ability to transfer shares can be blocked by anybody by 0 amount transferFrom call

type: code fragility / severity: minor

`HATVault.withdraw` implementation is confusing and potentially incorrect. Especially rounding up when converting to assets here seems problematic: HATVault.sol#L532 Instead of the back and forth conversion of shares and amount, I’d recommend refactoring the code to a more simple and straightforward implementation with just one conversion.

status - fixed

Issue has been fixed and is no longer present in

https://github.com/hats-finance/hats-contracts/tree/309d0832fd5a778a04c38a0d74c9c7eee3589449/contracts