Hats Audit / Jun 2021

Auditor: Adam Levi

Preface

First, we would like to thank Hats DAO for the pleasure of auditing their code. And we hope to further collaborate with them in the future.

General Overview

The Hats protocol is designed to give white hats hackers the opportunity to gain more on their good behaviour and contribution. Trying to tilt the balance of incentives, and incentivizing more hackers to act responsively. It is doing so by letting projects publish on-chain bounties for their protocols, with committees in-charge of approving or rejecting claims. To further increase the efficiency of this model, the HAT token is introduced, to help bootstrap both ends in this two-sided market.

Files in scope

Following the files in the final commit for this audit in https://github.com/hats-finance/hats-contracts/tree/be3fa22a871eb666f532d9c4224e56bfc10ef426

```
contracts/
    Governable.sol
    HATMaster.sol
    HATToken.sol
    HATVaults.sol
    tokenlock/
        CloneFactory.sol
        HATTokenLock.sol
        ITokenLock.sol
        ITokenLockFactory.sol
        MathUtils.sol
        OwnableInitializable.sol
        TokenLock.sol
        TokenLockFactory.sol
```

Current status

No major issues were found. As of July 2nd, all the smaller issues that were found have been fixed by the Hats team.

Methodology and Scope

We have extensively read the code and documentation of the system, examining different attacking vectors. Where prior code was used we have carefully reviewed both the original code and the new one, with emphasis on the changes.

For coherency we are slicing the scope of the audit into four major parts: 1. Token - `HATToken.sol` 2. Vesting - `HATTokenLock.sol` & `TokenLockFactory.sol` 3. Pools - `HATMaster.sol` 4. Vaults - `HATVaults.sol`

We will review each of them.

Issues by part

Part 1 - Token

The `HATToken.sol` contract is based on the Uni Token, which is well battle-tested. This is a very good practice.

Issue 1.a type: usability / severity: medium

In `confirmGovernance` and `confirmMinter` the delay time is defined in blocks. As the delay suppose to be in the scale of hours/days, this can cause very bad UX. Measuring with blocks is good to avoid small time-shifts by miners. But for long times can cause major time skews from what was expected, due to block time changes.

status - fixed

Issue has been fixed in https://github.com/hats-finance/hats-contracts/pull/49/

Part 2 - Vesting

The vesting contracts are based on TheGraph's vesting contracts, which are well battle-tested.

Issue 2.a type: readability / severity: very-low

In `TokenLock.sol`, the constructor is redundant and confusing.

status - fixed

Issue has been fixed in https://github.com/hats-finance/hats-contracts/pull/37

Part 3 - Pools

The pools contracts are based on bAlpha pools.

Issue 3.a type: readability / severity: low

In `HATMaster.sol`, the implementation of the getMultiplier with arrays was very inefficient and open for trouble.

status - fixed

Issue has been fixed in https://github.com/hats-finance/hats-contracts/pull/32

Part 4 - Vaults

The vaults are based on the pools, this architecture is used since the bAlpha reference was used. It creates a strange slicing between the two which requires attention.

Notes

- Note that approving a token by the governance requires checking that the Token contract is not adversarial.
- The users/pools must be aware that the protocol assumes that the governance is a trusted entity. As it has the power to do many things, e.g. drive the price up in Uniswap via a flash loan, trigger `swapBurnSend` making the contract buy Hats at a high price. Sell back the Hat tokens and clost the loan with profit.
- Uniswap LPs can track this contract, and remove liquidity before large swaps, which might create large slippage. This however is a general comment for every contract swaping on uniswap, and probably cannot be avoided.

Issue 4.a type: readability / severity: low

In `HATVaults.sol`, `setCommittee` was also used for committee check in, which was a bit confusing.

status - fixed

Issue has been fixed in https://github.com/hats-finance/hats-contracts/pull/44