Hidden Hand

Contracts Audit

March 2022

Task

We were tasked with performing a Smart Contracts security audit for the core business logic of “Hidden Hand”

Mission

The intention was to have an initial review of the contracts to identify:

- Architecture problems
- Code issues
- Suggestion of improvements

Code

ccfe90ffbdaf

The commit hash audited was:
d91a9a7e9050094d13c57b668d91

Summary

Each protocol has an associated contract for making bribes. Bribes are associated with proposals, which can only be added by “team members”. Each proposal has a deadline and a reward identifier is for rewards with an associated deadline. The funds are deposited into a vault contract by the bribe contracts. Admin can transfer the funds to a distributor contract. The receivers of rewards can get them by posting a merkle proof. The actual data about rewards is posted publicly online (offchain). From this data anyone can verify that the admin has worked correctly. (Another possibility is that there’s a deterministic algorithm that can compute this data, then the merkle root (and other hashes) that were posted by admin can be checked even if the data is not posted publicly)

We realized a static analysis with mythril and slyther.

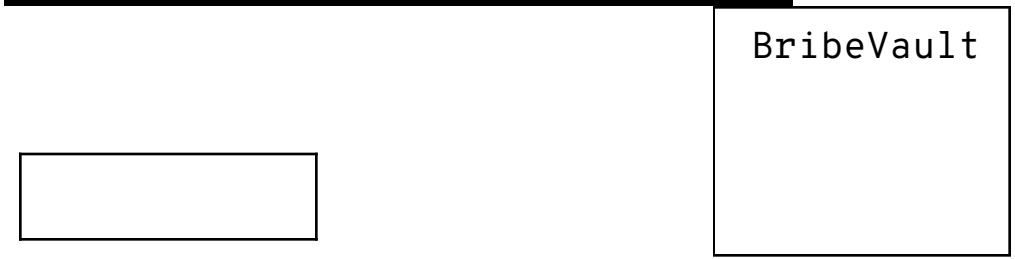
- Mythril: “control flow depends on block.timestamp”

- Slither complains about re-entrancy and style issues

We realized a manual review of the contracts about the following topics:

- BribeVault
- BribeBase
- TokemakBribe
- RibbonBribe
- LiquidDriverBribe
- FraxBribe
- RewardDistributor

Issues Found



depositBribeERC20 and transferBribes re-entrancy

● If erc 20 transfer somehow triggers another bribe, the amount in the internal transaction would be added twice

list length is limited by number of proposals with same deadline

● Only team members can add proposals and tokens, so one of them would have to be compromized

depositBribeERC20 and transferBribes should have re-entrancy protection

`BribeVault` calculateTransferAmounts: lists in rewardToBribes have no limit in length, might cause DoS by giving too many bribes to one reward identifier

It would be safer if there was some upper limit…

● Reward identifier includes token and deadline, so the

Issues Found

`BribeVault` BribeVault.transferBribes: it is possible to transfer before deadline …

- After that, there can still be more bribes deposited
- Only way to recover funds would be to use emergency withdrawal
- This might cause problems if the transfer is automated

- The problem is that then there might not be enough funds to transfer other rewards
- Probably the only case where this might cause problems is if the code is reused somewhere…

when transfer has happened, deposit should be impossible

`BribeVault` Fee max: fee should never be >= 50%

- otherwise distributorAmount in BribeVault.transferBribes might be zero even though something is transferred

there could be a comment warning about this edge case, or use distributorAmount + feeAmount

Issues Found

`RibbonBribe` `LiquidDrive` `rBribe` `FraxBribe`

● Methods where it could be checked that argument lists have same length: LiquidDriverBribe.setGaugeProposals, RibbonBribe.setGaugeProposals
● unused method: BribeBase._setProposals

Style issues

Why isn’t PROTOCOL

hardcoded in these contracts?

● initialize variables


Contact & Info

www.keyko.io

Keyko GmbH
Rote Trotte 9
CH-6340 Baar Zug
Switzerlan

Email:

Web:

Address:

info@keyko.io


keyko