Security assessment and code review

January 11 , 2022
Updated: Jan 25 - March 4, 2022

Prepared for:
Vovo.Finance

Prepared by:
Mikerah Quintyne-Collins | HashCloak Inc

Table Of Contents

Executive Summary

Vovo Finance engaged HashCloak Inc for an audit of their Vovo Finance smart contracts, specifically, their PrincipleProtectedVault, written in Solidity. The audit was done with one auditor over a 1 week period with subsequent smaller reviews done between January and March 2022.
The scope of the audit were solidity contracts in the **interfaces** and **products** folders. The commit hashes that were audited are the following:

- 014c7049ba653e83952753f599bb3c645984c786
- 369b0c46b2d20691407512e3cf3d6d11edaf6276
- 9631cf94b8f26ecc00c0ad2a592a8b2649da9aad
- 28e2ab5f4c66d4ca69e454eed70206506de7b3d9
- E0cde73edf7aa041b64e6a69a2a6c05247c79ef1
- E02ff3e3d550328524d4f4014c95cbc82be7006b
- F075fa00c637324299ffaece8d1df6ab069d5d74
- 1375621ce47b1bd9dcd385c802225ed39cece2d0

Throughout the course of the audit, we familiarized ourselves with the key functionality of the PrincipleProtectedVault contract and its key dependencies such as Curve, Uniswap and GMX. We manually checked the contract against well-known Solidity vulnerabilities relating to the following:

- Appropriate access control
- Unsafe arithmetic errors such as overflows, underflows or rounding errors
- DeFi-related attacks such as flash loans
- Unauthorized transfer of funds
- Sufficient code and test coverage
- Anything else as identified during the initial analysis phase

Further, we ran the PrincipleProtectedVault contract in off-the-shelf vulnerability detection tools Mythril and Slither. Mythril returned no results. Slither returned several results that we deemed to be false positives.

We initially identified issues ranging from low severity to informational and provided recommendations to improve code quality and mitigations against several attacks. After discussing with the Vovo.finance team, we updated our findings and only have an

informational issue that has been addressed. We also provide some general recommendations for further improving the code base.

Severity	**Number of Findings**
Critical	0
High	0
Medium	0
Low	0
Informational	1

Overview

Vovo Finance is a financial protocol that offers structured products on Ethereum and its L2s. Structured products are pre-packaged investments that can include different financial products such as derivatives and other assets. Vovo Finance makes use of GMX, a perpetual trading protocol, Curve, a stablecoin AMM and Uniswap V3, a general AMM.

Vovo Finance has a set of vaults which pools users' funds and manages the accounting of such funds. The main contract that contains all the functionality is the PrincipleProtectedVault contract. A vault owner can set a variety of parameters depending on their targeted risk profile such as setting the underlying token for the vault, setting the liquidity provider token and setting the addresses for the correct GMX, Curve and Uniswap contracts. Users deposit their assets, namely the underlying vault token and receive shares of the vault. They are rewarded periodically through calls to the earn and poke functions which are called by Keepers handled by the Vovo Finance team. Once a user is ready to withdraw, they are able to withdraw without permission. The earn function regularly deposits the liquidity held in the vault into Curve and deposits the corresponding LP tokens into Gauge, a Curve concept used for calculating how much inflation is going to users providing liquidity, in our case, the vault in question. Then, the poke function is called regularly to collect rewards from Curve, close old positions that were opened at GMX, swaps the reward to the underlying token of the vault and reinvests the rewards and profits into GMX and Curve.

Scope

The main scope of the audit was the PrincipleProtectedVault.sol contract in contracts/products. We also looked at the contract interfaces in contract/interfaces in order to get familiar with the APIs used to call external DeFi applications used within the PrincipleProtectedVault contract.

Findings

Potential for Re-entrancy in deposit and withdrawal functions

Type: Informational
Files affected: PrincipleProtectedVault.sol

In `deposit()`, `withdraw()`, `withdrawToVault()`, `withdrawAll()` are susceptible to a re-entrancy attack that may lead to events being emitted out of order. In particular, if the vault charges fees, the call on line 360 may lead to events being out of order if it re-enters. However, as the `_rewards` address is set by the governor of the vault, the governor should ensure that the contract at the `_rewards` address doesn't re-enter the vault. It should be mentioned that among the withdrawal functions in the PrincipleProtectedVault contract, that due to the access control checks on adding vaults that can be withdrawn to, `withdrawToVault()` is the least susceptible to this attack. In the case of `deposit()`, we believe that a re-entrancy attack would not enable a prospective attacker in fooling the contract to deposit more funds than what was deposited.

Impact: Events may appear out of order.
Suggestion: Apply the checks-and-effects pattern or a re-entrancy guard. Due to the nature of the implementations of the affected functions, we specifically recommend applying a re-entrancy guard.
Status: A re-entrancy guard has been applied at commit [9631cf94b8f26ecc00c0ad2a592a8b2649da9aad](9631cf94b8f26ecc00c0ad2a592a8b2649da9aad).

General Recommendations

Check return values of external calls

This codebase makes heavy use of external contracts and as such makes many external calls. We highly recommend that the return values of external calls are checked consistently throughout the codebase in order to fail gracefully and properly handle exceptions and reverts.