CERTIK

Security Assessment

HodlTree

Jun 16th, 2021

Table of Contents

Summary

This report has been prepared for HodlTree smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Majority of the findings are of informational nature related to gas optimization with one minor finding. The minor finding comprise the lack of input validation for function parameter. All of findings are remediated by the HodlTree team.

Overview

Project Summary

Project Name	HodlTree
Description	The audited codebase comprise the Liquidity Pool contract that allows depositing of specific set of tokens and in return get the liquidity tokens. These liquidity tokens later can be used to withdraw any of the tokens from the set provided that the withdrawer posses enough liquidity tokens. The withdrawal can be performed by either burning liquidity tokens and getting all the tokens in equal ratios based on the amount of burned liquidity tokens, specifying liquidity amount and using percentages of it for different withdrawing tokens or explicitly providing the withdrawing amounts and then burning liquidity tokens based on it. The contract also allows user to borrow the amounts from set of tokens as flash loan.
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/HodlTreeProtocol/stableFlashloan/tree/8f545faafff9c636311fca4193a69400e7f7ac04
Commit	8f545faafff9c636311fca4193a69400e7f7ac04, 3a9f3e86d08d57c1aeea4e28b2aa35794c160490

Audit Summary

Delivery Date	Jun 16, 2021
Audit Methodology	Manual Review
Key Components	

Vulnerability Summary

Total Issues	5
Critical	0
Major	0
Medium	0
Minor	1
Informational	4
Discussion	0

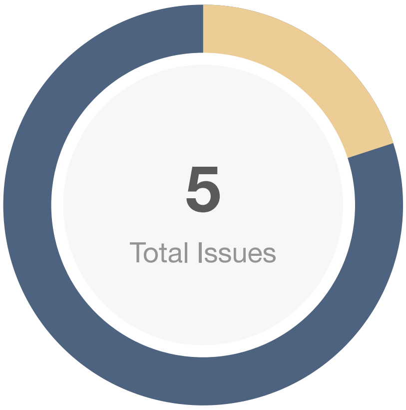
Audit Scope

ID	file	SHA256 Checksum
LPF	contracts/LiquidityPool.sol	e29f41330a4cd63ad8c5d21cc3341944447beed798144fd71cc083b1b19fc4e1

Executive summary

All findings were remediated.

Findings



- Critical **0** (0.00%)
- Major **0** (0.00%)
- Medium **0** (0.00%)
- Minor **1** (20.00%)
- Informational **4** (80.00%)
- Discussion **0** (0.00%)

ID	Title	Category	Severity	Status
LPF-01	Unlocked Compiler Version	Language Specific	Informational	Resolved
LPF-02	Lack of input validation	Volatile Code	Minor	Resolved
LPF-03	Missing error message	Coding Style	Informational	Partially Resolved
LPF-04	Inefficient storage read	Gas Optimization	Informational	Resolved
LPF-05	`require` statements can be substituted with modifier	Coding Style	Informational	Resolved

LPF-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	contracts/LiquidityPool.sol: 2	Resolved

Description

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.7.6` the contract should contain the following line:

```
pragma solidity 0.7.6;
```

Alleviation

Issue has been resolved.

LPF-02 | Lack of input validation

Category	Severity	Location	Status
Volatile Code	Minor	contracts/LiquidityPool.sol: 96	Resolved

Description

The function `setAdminFeeAddress` does not validate its address type parameter which can result in unwanted state of the contract if it passed as a zero address.

Recommendation

We would advise to add input validation by having a require statement checking against zero address.

Alleviation

Issue has been resolved.

LPF-03 | Missing error message

Category	Severity	Location	Status
Coding Style	Informational	contracts/LiquidityPool.sol: 205, 233, 256, 344	Partially Resolved

Description

The linked require statements do not specify error messages.

Recommendation

We recommend to add error messages to aforementioned require statements to increase the legibility of codebase.

Alleviation

Issue partially resolved. Line 233 is still missing an error message in require

LPF-04 | Inefficient storage read

Category	Severity	Location	Status
Gas Optimization	● Informational	contracts/LiquidityPool.sol: 219, 224, 236	Resolved

Description

The aforementioned lines read `adminBalance` from contract's storage which results in increased gas cost.

Recommendation

We advise to introduce a local variable and initialize it with `adminBalance` before using it on the aforementioned lines to save gas cost associated with additional storage reads.

Alleviation

Issue has been resolved.

LPF-05 | `require` statements can be substituted with modifier

Category	Severity	Location	Status
Coding Style	● Informational	contracts/LiquidityPool.sol: 116, 126	Resolved

Description

The `require` statements on the aforementioned lines can be substituted with modifier to increase the legibility of codebase.

Recommendation

We advise to substitute `require` statements on the aforementioned lines with modifier. The `require` statements can wrapped inside a function before being called from modifier to reduce the contracts bytecode footprint.

Alleviation

Issue has been resolved.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK