CERTIK

Security Assessment

LiquityVault

Apr 22nd, 2021

Summary

This report has been prepared for LiquityVault 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 Dynamic Analysis, Static Analysis, and 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.

Addtionally,this audit is based on a premise that external smart contracts were implemented safely.

The security assessment resulted in 6 informational findings. 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.

Overview

Project Summary

Project Name	LiquityVault
Description	Liquity is a decentralized borrowing protocol
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/Liquityfi/LiquityVault
Commits	412de290f7fc071312f24f73b41541455fecc535

Audit Summary

Delivery Date	Apr 22, 2021
Audit Methodology	Manual Review
Key Components	

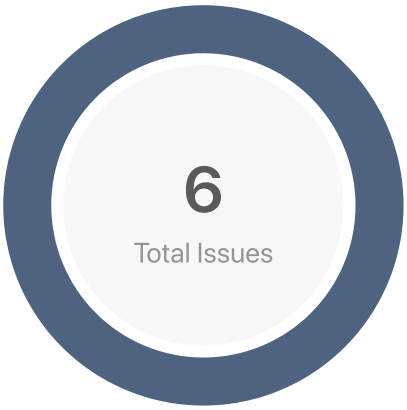
Vulnerability Summary

Total Issues	6
Critical	0
Major	0
Minor	0
Informational	6
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
CLV	Converter.sol	88b9c106b3eef5949f3a9b84d90d50543fd5f0ebcc9a4a04979da2e893959d35
SDA	SaftDAO.sol	b14acaa553b4e2b041311de19801cbac3de9d7235bc47a5ce13f1e375e9ee8a4
SCV	StableCoinsVault.sol	c85548a89dcda3c38c39e42085a57b5d1c2c151bc9e03b7739f345161d0152bc
TLV	Timelock.sol	703f24033b597d6d9fade36f4252bbc63c805145ab2f580247b5d139d02c0a15

Findings



Critical	0 (0.00%)
Major	0 (0.00%)
Minor	0 (0.00%)
Informational	6 (100.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
CLV-01	Zero Address in Converter.sol	Logical Issue	Informational	Partially Resolved
CLV-02	Proper Usage of "public" And "external" Type	Gas Optimization	Informational	Resolved
SCV-01	Zero Address in StabelCoinsVault.sol	Logical Issue	Informational	Partially Resolved
SCV-02	Discussion About The _withdraw Function	Logical Issue	Informational	Acknowledged
SCV-03	Missing Emit Event	Logical Issue	Informational	Resolved
SDA-01	Proper Usage of "public" And "external" Type	Gas Optimization	Informational	Resolved

CLV-01 | Zero Address in Converter.sol

Category	Severity	Location	Status
Logical Issue	Informational	Converter.sol: 42~50	Partially Resolved

Description

The `address` value should be verified as non zero value to prevent being mistakenly assigned as `address(0)` .

Recommendation

Check that the address is not zero by adding following checks in the contract `Converter.sol`. For example:

```
function setPath(
        address _token0,
        address _token1,
        uint8 _swapType,
        address _swapRouter,
        address[] memory _uniSwapPath,
        int128[] memory _crvSwapPath
    ) public onlyOwner {
        require(_token0!=address(0),"_token0 is a zero address!");
        require(_token1!=address(0),"_token1 is a zero address!");
        require(_swapRouter!=address(0),"_swapRouter is a zero address!");
    }
```

Alleviation

The team heeded some of our advice and changed related codes. Code change was applied in commit 3f120b0c5b6ea7992adf7ce45d5e6750f1860290.

CLV-02 | Proper Usage of "public" And "external" Type

Category	Severity	Location	Status
Gas Optimization	Informational	Converter.sol: 80	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. `external` function costs less gas than `public` function.

Recommendation

Considering using the `external` attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 3f120b0c5b6ea7992adf7ce45d5e6750f1860290.

SCV-01 | Zero Address in StabelCoinsVault.sol

Category	Severity	Location	Status
Logical Issue	Informational	StableCoinsVault.sol: 74, 96, 100, 111, 115	Partially Resolved

Description

The `address` value should be verified as non zero value to prevent being mistakenly assigned as `address(0)` .

Recommendation

Check that the address is not zero by adding following checks in the functions of contract `StableCoinsVault.sol`. For example:

```
        function setFeeTo(address _feeTo) external onlyOwner {
            require(_feeTo!=address(0),"_feeTo is a zero address!");
        ...
    }
```

Alleviation

The team heeded some of our advice and changed related codes. Code change was applied in commit 3f120b0c5b6ea7992adf7ce45d5e6750f1860290.

SCV-02 | Discussion About The _withdraw Function

Category	Severity	Location	Status
Logical Issue	Informational	StableCoinsVault.sol: 220~226	Acknowledged

Description

In the _withdraw function, when the `LUSD` balance of the contract account is insufficient, the difference of `LUSD` will be withdrawn from `SP`. If it is still insufficient, the user's withdrawn `LUSD` will be less than the invested `LUSD`. Will the user lose asset?

Alleviation

Customer response: Generally, the vault can withdraw enough assets from SP pool. If not, it means SP pool meets some problems. In such a case, the withdrawn amount means how many assets the vault protects for users.

SCV-03 | Missing Emit Event

Category	Severity	Location	Status
Logical Issue	Informational	StableCoinsVault.sol: 92, 96, 100, 106, 111, 115	Resolved

Description

Some functions should be able to emit event as notifications to customers because they change the status of sensitive variables.This suggestion is not limited to these codes, but also applies to other similar codes.

Recommendation

Consider adding an emit after changing the status of variables.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 3f120b0c5b6ea7992adf7ce45d5e6750f1860290.

SDA-01 | Proper Usage of "public" And "external" Type

Category	Severity	Location	Status
Gas Optimization	Informational	SaftDAO.sol: 112, 120, 126, 132, 157, 171, 190, 221, 223	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. `external` function costs less gas than `public` function.

Recommendation

Considering using the `external` attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 3f120b0c5b6ea7992adf7ce45d5e6750f1860290.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that 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.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

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

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in storage one.

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 and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

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.