



Security Assessment

PolyQuity

Jun 28th, 2021



Table of Contents

Summary

Overview

Project Summary

Audit Summary

Vulnerability Summary

Audit Scope

Findings

MSL-01 : Unused state variable

MSL-02 : State variable could be declared as immutable

MSL-03 : Missing parameter check

PFP-01 : Unused state variables

PFP-02 : Redundant judgment

Appendix

Disclaimer

About

Summary

This report has been prepared for PolyQuity 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 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.

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.

Overview

Project Summary

Project Name	PolyQuity
Description	PolyQuity is committed to Liquity's vision of providing a decentralized borrowing protocol with interest-free loans, high capital efficiency, and censorship-resistant stable coins.
Platform	Polygon
Language	Solidity
Codebase	https://github.com/PolyQuity/contracts
Commit	5d66b2144c8c89a862577eb2e62bae6f3baccffa 56422e09619d6fef89f52824a5e6619132ec95f3 15906354b48d79abcd964de60c43cbd1f2581c1f

Audit Summary

Delivery Date	Jun 28, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total Count	Pending	Partially Resolved	Resolved	Acknowledged	Declined
● Critical	0	0	0	0	0	0
● Major	0	0	0	0	0	0
● Medium	0	0	0	0	0	0
● Minor	1	0	0	1	0	0
● Informational	4	0	0	3	1	0
● Discussion	0	0	0	0	0	0

Audit Scope

ID	file	SHA256 Checksum
LQT	LPRewards/LQTYUnipool.sol	a42bc9b3f6263516b97152eab79f42ad9959f8edb01e98d51bfe04eef50a01ba
ULP	LPRewards/Unipool.sol	34ef42c58b50aa7d0f06ae27181eb974a49e5fb522f5d62674c7bf60150f5434
LQL	LQTY/LQTYToken.sol	ea4ead8cda3809b2a0c3fabb54a2beee0b4e9331dbf25bdbfb5123bff7f4d58f
MSL	LQTY/MultiSig.sol	b984f14b7e993a73ae93bca37a9bca2bf3ec86d87439e02af6711a180083fe41
PFP	PriceFeed.sol	8bc99b5617f301c9623c8fbd771ea7dac2a056f2241ad2788b9aee994193ee22

Review Summary

PolyQuity is a decentralized borrowing protocol on Polygon. The scope of this audit is a part of the whole protocol. This time CertiK audited the following five contracts:

- Unipool - Liquidity providers stake the UNIV2 LP tokens into `Unipool` to accrue rewards.
- LQTYToken - An ERC20 token based on the `Openzeppelin ERC20` contract.
- LQTYUnipool - Similar to `Unipool` contract. Add some functions to stake LQTY token.
- MultiSig - This contract is multi-signature. this contract makes it possible for two or more owners to sign approval as a group.
- PriceFeed - PriceFeed for mainnet deployment, to be connected to Chainlink's live `MATIC:USD` aggregator reference contract, and a wrapper contract `BandCaller`, which connects to `Band` contract.

CertiK analyzed and conducted the source code through a variety of methods and tools, such as CertiK Formal Verification as well as manual review by smart contract experts.

At the moment, the PolyQuity team did not have testing and documentation repositories available for reference. CertiK recommends additional unit test coverage, along with documentation, to more thoroughly simulate potential use cases and functionalities for token holders. CertiK always recommends seeking multiple opinions, increased test coverage, and live sandbox deployments before a mainnet release.

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
MSL-01	Unused state variable	Coding Style	● Informational	✓ Resolved
MSL-02	State variable could be declared as immutable	Gas Optimization	● Informational	✓ Resolved
MSL-03	Missing parameter check	Logical Issue	● Minor	✓ Resolved
PFP-01	Unused state variables	Coding Style	● Informational	✓ Resolved
PFP-02	Redundant judgment	Logical Issue	● Informational	ⓘ Acknowledged

MSL-01 | Unused state variable

Category	Severity	Location	Status
Coding Style	● Informational	LQTY/MultiSig.sol: 16	✓ Resolved

Description

State variable `decimals` is never used in contract `MultiSig`.

Recommendation

We recommend that remove the variables never used.

Alleviation

[PolyQuity]: State variables `decimals` is a standard ERC20 variable. We hope this MultiSig contract would be compatible with ERC20 token.

MSL-02 | State variable could be declared as immutable

Category	Severity	Location	Status
Gas Optimization	● Informational	LQTY/MultiSig.sol: 21	✓ Resolved

Description

Private variable `tokenSetter` only set value in the constructor function. Declare it as `immutable` would save gas.

Recommendation

We recommend that declare the variable as `immutable` to save gas.

Alleviation

PolyQuity added `immutable` attribute in commit `56422e09619d6fef89f52824a5e6619132ec95f3`.

MSL-03 | Missing parameter check

Category	Severity	Location	Status
Logical Issue	● Minor	LQTY/MultiSig.sol: 72~75	✓ Resolved

Description

Function `addOwner` is used to add a new owner address to the `hasSignAddOwner` mapping. The new owner address should not be zero address.

Recommendation

We recommend that add a zero check for the `_newOwner` parameter.

Alleviation

PolyQuity added a zero address check for `_newOwner` in commit `56422e09619d6fef89f52824a5e6619132ec95f3`.

PFP-01 | Unused state variables

Category	Severity	Location	Status
Coding Style	● Informational	PriceFeed.sol: 33~36	✓ Resolved

Description

State variables `borrowerOperationsAddress`, `troveManagerAddress`, and `ETHUSD_TELLOR_REQ_ID` are never used in contract `PriceFeed`.

Recommendation

We recommend removing the variables never used.

Alleviation

PolyQuity removed state variable `ETHUSD_TELLOR_REQ_ID` and kept the others. This change is applied in commit `56422e09619d6fef89f52824a5e6619132ec95f3`.

PFP-02 | Redundant judgment

Category	Severity	Location	Status
Logical Issue	● Informational	PriceFeed.sol: 502	① Acknowledged

Description

The `else if` condition is redundant.

Recommendation

We recommend that remove the redundant `else if` code.

Alleviation

[PolyQuity]: Our protocol is forked from Liquity which is audited already. We would not change the code to keep similar with the Liquity.

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.

Logical Issue

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

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.

