Security Assessment

Moonlift

Jun 2nd, 2021

Table of Contents

Summary

This report has been prepared for Moonlift 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	Moonlift
Description	A community driven passive income generation protocol.
Platform	BSC
Language	Solidity
Codebase	https://github.com/moonlift/moonlift
Commits	<1087aa50a15f02d214620e9a034927092b9a6d88>

Audit Summary

Delivery Date	Jun 02, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

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

Audit Scope

ID	file	SHA256 Checksum
MOO	Moonlift.sol	7e93200b4c5379c8283a28b36b9f9c0e5b350c21be4b05019a0749897aed931c
BEP	abstracts/BEP20WithFee.sol	1b997c8f40d1efa360f572a24adce52de5bdf7877ab98cefca2048b35430ae0d
BEW	abstracts/BEP20WithHoldersDistribution.sol	cc819a001e0ffa788d7646cada1ce7cf5794912f6e1dc4837e6d7338a50abf2e
GOV	abstracts/Governance.sol	1b89fa71c680d656a3a055fd9df0ff04a477ff6edf63794b0580a4dfee0909b3
PHT	abstracts/PairsHolder.sol	d8b443d0845b6498d53f0b85499716fc25f35d0571583d2f59f4e037a67c76b9
IUV	interfaces/IUniswapV2Factory.sol	4382b6dd8167100b1865eeaeb6deae8a3f4df35c0511e8cf49b16a5e7b972480
IUP	interfaces/IUniswapV2Pair.sol	d2969d170bfeda3d102d30b4cad0aea8f2b5b1b19fc1d1e331c3f33ab02cf86e
IUR	interfaces/IUniswapV2Router01.sol	2193ce49961563eff49b5f5cf4d8b9dcb1ce7a12aee1ecb26e8ee52637db1578
IVR	interfaces/IUniswapV2Router02.sol	865501a5fbfdb23cf916e742889ac70a3dc7534507d15f2f58334688d86c4969
ADD	libs/Address.sol	3169398ee2562be235d9ed86ea9eb3140abdaca4d92c62937a98f49f8def68cf
BE2	libs/BEP20.sol	a78c5655bbe0314a1d1b799bbdb15b6df1a9573f1e5d41064c662ce577399f68
CON	libs/Context.sol	1f4e92235e59e0894b64554bf75e450baf9f14bc7326a6916eb24662e9ef3727
IBE	libs/IBEP20.sol	b4b340bc68a288129f27ba68dfa3da24a73847c1ada4e05bf8ff14d16490351d
OWN	libs/Ownable.sol	e4a22a8a74f38620ac43b7f4479cb176204f31018fb819adb18beb8d994281c2
SBE	libs/SafeBEP20.sol	675b45e95ddb74b5dc4075aee911925f3cc1de8722be9db1ebb6f072d20af058
SMT	libs/SafeMath.sol	652c344aa21df17d313aa6ad0b3294e78efa9896d224b9b88b345263e18a994e

Findings

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

ID	Title	Category	Severity	Status
BE2-01	Different pragma versions	Coding Style	Informational	Acknowledged
BEP-01	State variable could be declared constant	Gas Optimization	Informational	Acknowledged
BEP-02	Missing emit event	Control Flow	Informational	Acknowledged
BEP-03	Constant value could be declared as a constant variable	Coding Style	Informational	Acknowledged
BEW-01	Use "SafeMath"	Mathematical Operations	Informational	Acknowledged
MOO-01	Missing parameter check	Gas Optimization	Informational	Acknowledged

BE2-01 | Different pragma versions

Category	Severity	Location	Status
Coding Style	Informational	libs/BEP20.sol: 2	Acknowledged

Description

This protocol uses some different solidity versions, such as `pragma solidity >=0.6.12;` and `pragma solidity 0.6.12;`. This is not recommended. Pragmas should be locked to a specific compiler version and flag that it has been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler, which may have higher risks of undiscovered bugs.

Recommendation

Avoid a floating pragma version instead specify pragma version without using the caret symbol, i.e. `pragma solidity 0.6.11;`

Deploy with any of the following solidity versions:

- 0.5.11 - 0.5.13
- 0.5.15 - 0.5.17
- 0.6.8
- 0.6.10 - 0.6.11

We recommend using a simple pragma version that allows any of these versions.

Alleviation

[Moonlift Team]: That is included library and it is taken as is to be verifiable.

BEP-01 | State variable could be declared constant

Category	Severity	Location	Status
Gas Optimization	Informational	abstracts/BEP20WithFee.sol: 23	Acknowledged

Description

The state variable `minLPBalance` is never be changed in the contract. It should be declared constant to save gas. And constant variable should be named UPPER_CASE_WITH_UNDERSCORES.

```
23    uint256 private minLPBalance = 1e18;
```

Recommendation

We recommend declaring the variable `minLPBalance` constant, and naming it UPPER_CASE_WITH_UNDERSCORES.

Alleviation

[Moonlift Team]: It is declared as constant. So just a style guide for the naming and not gas optimization.

BEP-02 | Missing emit event

Category	Severity	Location	Status
Control Flow	Informational	abstracts/BEP20WithFee.sol: 79, 83, 87	Acknowledged

Description

Several sensitive actions are defined without event declarations.

Recommendation

We recommend adding events for the sensitive actions, and emit them in the functions.

Alleviation

[Moonlift Team]: Agree. There could be additional events emited, but that was not planned.

BEP-03 | Constant value could be declared as a constant variable

Category	Severity	Location	Status
Coding Style	Informational	abstracts/BEP20WithFee.sol: 64~66, 134, 138, 198~201	Acknowledged

Description

The value `10000` is used many times in the contract. It is better to declared constant variable to save gas.

Recommendation

We recommend declaring the value `10000` as a constant variable.

Alleviation

[Moonlift Team]: It is a coding style suggestion.

BEW-01 | Use "SafeMath"

Category	Severity	Location	Status
Mathematical Operations	Informational	abstracts/BEP20WithHoldersDistribution.sol: 24	Acknowledged

Description

Function `_calcRewards` does not use `SafeMath`.

Recommendation

We recommend using `SafeMath` for calculations.

Alleviation

[Moonlift Team]: `SafeMath` should be used for calculations, but right here we are never getting under or overflow issues in these calculations, so here `SafeMath` would be an overhead.

MOO-01 | Missing parameter check

Category	Severity	Location	Status
Gas Optimization	Informational	Moonlift.sol: 42, 46	Acknowledged

Description

Function `burn` and `burnFrom` parameters are missing greater than zero checks.

Recommendation

We recommend adding parameter checks in the functions to save gas.

Alleviation

[Moonlift Team]: Agree. That could save a little gas on 0-burn transactions if anyone try to burn 0.

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.

Mathematical Operations

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

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.

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.