HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: LaunchZone
Date: November 10th, 2021

This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed — upon a decision of the Customer.

Document

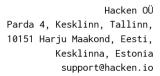|	
Name	Smart Contract Code Review and Security Analysis Report for LaunchZone.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Staking
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/ezDeFi/sfarm-contracts
Commit	030dbebb3a8a7e0e93dc6382a103757b07a912ff
Technical Documentation	NO
JS tests	YES
Timeline	17 SEPTEMBER 2021 - 10 NOVEMBER 2021
Changelog	24 SEPTEMBER 2021 - Initial Audit 07 OCTOBER 2021 - Second Review 10 NOVEMBER 2021 - Third Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by LaunchZone (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between September 17th, 2021 - September 24th, 2021.

Second review conducted on October 7th, 2021.

Third review conducted on November 10th, 2021.

Scope

The scope of the project is smart contracts in the repository:
Repository:
    https://github.com/ezDeFi/sfarm-contracts
Commit:
    030dbebb3a8a7e0e93dc6382a103757b07a912ff
Technical Documentation: No
JS tests: Yes
Contracts:
    interfaces\IERC20.sol
    interfaces\Initializable.sol
    interfaces\IUniswapV2Pair.sol
    interfaces\IUniswapV2Router01.sol
    interfaces\Upgradable.sol
    interfaces\ICitizen.sol
    lib\CapMath.sol
    lib\SafeMath.sol
    lib\StakeLib.sol
    lib\UniswapV2Library.sol
    lib\util.sol
    Bank.sol
    DataStructure.sol
    Migrations.sol
    Proxy.sol
    Role.sol
    Timelock.sol
    Token.sol

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.

Insecure	Poor secured	Secured	Well-secured

You are here

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **3** low severity issues.

After the second review security engineers found **5** low severity issues.

After the third review security engineers found **all** issues were addressed.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution

Audit overview

■ ■ ■ ■ **Critical**

No critical issues were found.

■ ■ ■ **High**

No high severity issues were found.

■ ■ **Medium**

No medium severity issues were found.

■ **Low**

1. Unresolved TODOs

   There are few TODOs in the source code. That may be assumed that the code is not fully written and some functionality is still unimplemented.

   Recommendation: Please implement all TODOs

   Status: Fixed.

2. Unused import statements

   There is an import in the code which isn't used in the contracts and just burn gas.

   Recommendation: Please remove unused imports

   Status: Fixed.

3. Maximum line length

   According to the [solidity style guide](): keeping lines under the PEP 8 recommendation to a maximum of 79 (or 99) characters helps readers easily parse the code.

   Recommendation: Please follow the [style guide]()
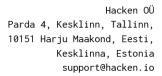
   Status: Fixed.

4. Reading the state variable in the loop

   Continuously reading the state variable in the loop is causing an unnecessary gas over-usage.

   Recommendation: To keep a little gas usage please store the state variable's value to the local variable.

   Status: Fixed.

5. A public function that could be declared external

public functions that are never called by the contract should be declared **external** to save gas.

Recommendation: Use the **external** attribute for functions never called from the contract.

Status: Fixed.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found **3** low severity issues.

After the second review security engineers found **5** low severity issues.

After the third review security engineers found all issues were addressed.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only – we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.