

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT



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 Sylo - Second Review
Approved by	Andrew Matiukhin CTO Hacken OU
Type	Staking
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Git repository	https://github.com/dn3010/staking-contracts/commit/06aa6826e6eaf4ebddb62c8fa4b3c1db8d0139e5 https://github.com/dn3010/staking-contracts/tree/598c6be88c4ff6f55904e0f98ff8d8a07fc0f717
Timeline	28 JUNE 2021 - 7 JULY 2021
Changelog	30 June 2021 - initial audit (06aa6826e6eaf4ebddb62c8fa4b3c1db8d0139e5) 7 July 2021 - second review (598c6be88c4ff6f55904e0f98ff8d8a07fc0f717)



Table of contents

Introduction	4
Scope	4
Executive Summary	6
Severity Definitions	7
Audit overview	8
Conclusion	10
Disclaimers	11

Introduction

Hacken OÜ (Consultant) was contracted by Sylo (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on July 7th, 2021.

Scope

The scope of the project is the smart contracts in Git Repository:

Repository:

<https://github.com/dn3010/staking-contracts/tree/598c6be88c4ff6f55904e0f98ff8d8a07fc0f717>
Staking.sol
YieldFarm.sol

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

Category	Check Item
Code review	<ul style="list-style-type: none">ReentrancyOwnership TakeoverTimestamp DependenceGas Limit and LoopsDoS with (Unexpected) ThrowDoS with Block Gas LimitTransaction-Ordering DependenceStyle guide violationCostly LoopERC20 API violationUnchecked external callUnchecked mathUnsafe type inferenceImplicit visibility levelDeployment ConsistencyRepository ConsistencyData Consistency



Functional review	<ul style="list-style-type: none">▪ Business Logics Review▪ Functionality Checks▪ Access Control & Authorization▪ Escrow manipulation▪ Token Supply manipulation▪ Asset's integrity▪ User Balances manipulation▪ Kill-Switch Mechanism▪ Operation Trails & Event Generation
-------------------	---

Executive Summary

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

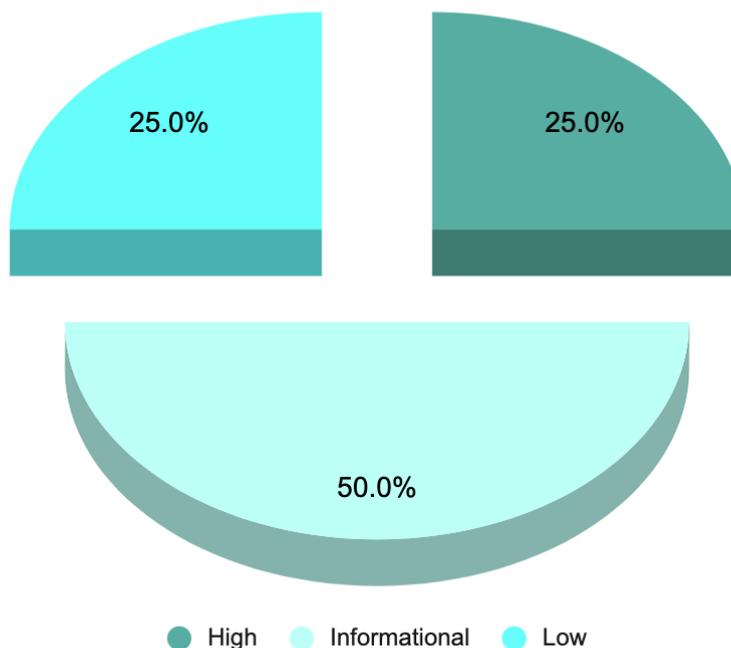


Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview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

Security engineers found 1 high, 1 low and 2 informational issues during the first review.

Security engineers found no issues during the second review.

Graph 1. The distribution of vulnerabilities after the first review.



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
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

Audit overview

■ ■ ■ ■ Critical

No Critical severity issues were found.

■ ■ ■ High

1. Vulnerability: Unchecked transfers in critical functions

ERC20 doesn't require a token to revert in the case of transfer/transferFrom failure, but it does require to return a boolean result of whether it succeeded.

Recommendation: Please consider adding a return value check.

Fixed before second review

■ ■ Medium

No Medium severity issues were found.

■ Low

1. Vulnerability: Missing zero address validation

The constructor of the YieldFarm contract doesn't check lpTokenAddress and communityVault addresses for a zero which can lead to unpredictable results.

Fixed before second review

■ Lowest / Code style / Best Practice

1. Vulnerability: Visibility for constructor

Visibility for constructor is ignored starting solidity version 0.7.0

Recommendation: Please consider removing visibility declaration from the constructor

Fixed before second review

2. Vulnerability: Public function that could be declared external



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

Recommendation: Please consider using the external attribute for functions never called from the contract

Fixed before second review



Conclusion

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

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

Security engineers found 1 high, 1 low and 2 informational issues during the first review.

Security engineers found no issues during the second review.



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 security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree 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 security of smart contracts.

Technical Disclaimer

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