

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT



Customer: Sylo

Date: December 3rd, 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 Sylo.		
Approved by	Andrew Matiukhin CTO Hacken OU		
Туре	ERC21 token; Staking; Micro-Payments; Epochs		
Platform	Ethereum / Solidity		
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review		
Repository	https://github.com/dn3010/sylo-ethereum-contracts		
Commit	8eec226ac5a7006ec97d5107a531f90b7aa9604e		
Technical	YES		
Documentation			
JS tests	YES		
Website	sylo.io		
Timeline	19 NOVEMBER 2021 - 03 DECEMBER 2021		
Changelog	29 NOVEMBER 2021 - INITIAL AUDIT 03 DECEMBER 2021 - SECOND REVIEW		

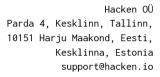




Table of contents

Introduction	
Scope	4
Executive Summary	6
Severity Definitions	7
Audit overview	8
Conclusion	12
Disclaimers	13



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 the Customer's smart contract and its code review conducted between November 19th, 2021 - November 29th, 2021.

Second review conducted on December 3rd, 2021.

Scope

```
The scope of the project is smart contracts in the repository:
Repository:
      https://github.com/dn3010/sylo-ethereum-contracts
Commit:
      8eec226ac5a7006ec97d5107a531f90b7aa9604e
Technical Documentation: Yes (<a href="https://github.com/dn3010/sylo-ethereum-">https://github.com/dn3010/sylo-ethereum-</a>
contracts/blob/master/docs/spec.md )
JS tests: Yes (https://github.com/dn3010/sylo-ethereum-
contracts/blob/master/test )
Contracts:
      Epochs/Manager.sol
      Payments
      Payments/Ticketing
      Payments/Ticketing/Parameters.sol
      Payments/Ticketing/RewardsManager.sol
      Payments/Ticketing.sol
      Staking
      Staking/Directory.sol
      Staking/Manager.sol
      ECDSA.sol
      Listings.sol
      Token.sol
      Utils.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	
Tunctional Teview	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 her	e 🚺

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 12 low severity issues.

After the second review security engineers found that 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. Too many digits

Literals with many digits are difficult to read and review.

Contracts: SyloToken.sol

Function: constructor

Recommendation: Please consider using scientific notation or separate zeros with underscores for clearity (ie: 10e9 or 10_000_000_000).

Status: Fixed

2. A public function that could be declared external

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

Contracts: EpochsManager.sol

Function: initialize, initializeEpoch, getEpoch

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

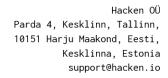
Status: Fixed

No event emitted

Changing significant values of the contract requires emitting an event to let the community the ability to track such changes off-chain.

Contracts: Parameters.sol

Function: setFaceValue, setBaseLiveWinProb, setExpiredWinProb, setDecayRate, setTicketDuration





Recommendation: Please emit an event when changing important values of the contract.

Status: Fixed

4. A public function that could be declared external

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

Contracts: Parameters.sol

Function: initialize, setFaceValue, setBaseLiveWinProb, setExpiredWinProb, setDecayRate

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

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

Contracts: RewardsManager.sol

Function: initialize, getLastClaim, getRewardPool, getRewardPoolStakersTotal, getRewardPoolActiveStake, getUnclaimedNodeReward, getUnclaimedStakeReward, initializeNextRewardPool, incrementRewardPool, calculateStakerClaim, claimStakingRewards, claimNodeRewards, addManager, removeManager

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

Status: Fixed

A public function that could be declared external

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

Contracts: Ticketing.sol

Function: initialize, redeem, withdraw, lockDeposits, unlockDeposits, depositPenalty, depositEscrow, setUnlockDuration

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

Status: Fixed

7. No event emitted



Changing significant values of the contract requires emitting an event to let the community the ability to track such changes off-chain.

Contracts: Ticketing.sol

Function: setUnlockDuration

Recommendation: Please emit an event when changing important values of

the contract.

Status: Fixed

8. A public function that could be declared external

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

Contracts: Directory.sol

Function: initialize, setCurrentDirectory, joinNextDirectory, scan,

getTotalStakeForStakee, getTotalStake, getEntries

Recommendation: Use the external attribute for functions never called

from the contract.

Status: Fixed

9. No event emitted

Changing significant values of the contract requires emitting an event to let the community the ability to track such changes off-chain.

Contracts: Directory.sol

Function: setCurrentDirectory

Recommendation: Please emit an event when changing important values of

the contract.

Status: Fixed

A public function that could be declared external

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

Contracts: Manager.sol

Function: initialize, setUnlockDuration, setMinimumStakeProportion, addStake, unlockStake, withdrawStake, cancelUnlocking, getTotalManagedStake, getStakeEntry,

calculateMaxAdditionalDelegatedStake



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

Status: Fixed

11. No event emitted

Changing significant values of the contract requires emitting an event to let the community the ability to track such changes off-chain.

Contracts: Manager.sol

Function: setUnlockDuration, setMinimumStakeProportion

Recommendation: Please emit an event when changing important values of

the contract.

Status: Fixed

12. A public function that could be declared external

public functions that are never called by the contract should be

declared **external** to save gas

Contracts: Listings.sol

Function: initialize, setListing, getListing

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 12 low severity issues.

After the second review security engineers found that 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.