HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Kalmar
Date: June 08th, 2021

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for Kalmar.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	dApps
Platform	Binance Smart Chain / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/kalmar-io/leverage-yield-contracts-busd
Commit	1FD562E807AE8269212D67ABD54275286875C43C
Timeline	28 MAY 2021 - 8 JUNE 2021
Changelog	8 JUNE 2021 - INITIAL AUDIT

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Kalmar (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 between 28th May, 2021 - 8th June, 2021.

Scope

The scope of the project is smart contracts in the repository:

```
Repository: https://github.com/kalmar-io/leverage-yield-contracts-busd
Commit: 1FD562E807AE8269212D67ABD54275286875C43C
```

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

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured. Some contracts with important functionality are not provided for the audit.

Insecure	Poor secured	Secured	Well-secured

You are here

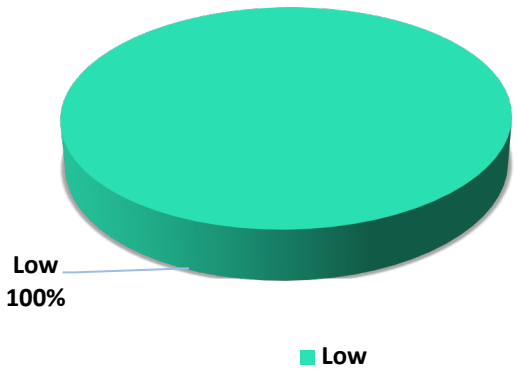
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview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

As a result of the audit, security engineers found **2** low severity issues.

Notices:

1. Description of contracts logic is not provided by the Customer and we may not prove correctness of some calculation.
2. The code is not covered with unit tests. We strongly recommend covering as much code as possible.

Graph 1. The distribution of vulnerabilities after the audit.



Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Audit overview

Critical

No critical issues were found.

High

No high severity issues were found.

Medium

No medium severity issues were found.

Low

1. safeApprove(address, uint256(-1)) function may fail for some specific implementation of underlying token. For example, COMP token volatiles the ERC-20 standard and reverts in if max uint256 is passed.

   Contracts: StrategyWithdrawMinimizeTrading.sol, StrategyLiquidate.sol, StrategyAllETHOnly.sol.

   Functions: execute

   Recommendation: ensure that lp pairs with such tokens are not added to the system.

2. `factory` field is never used.

   Contracts: MasterChefGoblin.sol, MasterChefPoolRewardPairGoblin.sol

   Recommendation: remove unused variables.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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.

As a result of the audit, security engineers found **2** low severity issues.

Notices:

1. Description of contracts logic is not provided by the Customer and we may not prove correctness of some calculation.
2. The code is not covered with unit tests. We strongly recommend covering as much code as possible.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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 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.