HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: KingDefi
Date: September 16th, 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 KingDefi.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Farming
Platform	Binance Smart Chain / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/KingDeFi-Solidity/SmartContractTests
Commit	862af556f50cb7917b479c1e22d5cf0c40703087
Technical Documentation	NO
JS tests	YES
Timeline	08 SEPTEMBER 2021 - 16 SEPTEMBER 2021
Changelog	10 SEPTEMBER 2021 - INITIAL AUDIT 14 SEPTEMBER 2021 - SECOND REVIEW 15 SEPTEMBER 2021 - THIRD REVIEW 16 SEPTEMBER 2021 - FOURTH REVIEW

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 KingDefi (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 September 8th, 2021 - September 10th, 2021.

Second review conducted on September 14th, 2021.

Third review conducted on September 15th, 2021.

Fourth review conducted on September 16th, 2021.

Scope

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

```
Repository:
      https://github.com/KingDeFi-Solidity/SmartContractTests
Commit:
      862af556f50cb7917b479c1e22d5cf0c40703087
Technical Documentation: No
JS tests: Yes
Contracts:
      KrownAutoCompoundFarm.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

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

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 1 high, 3 medium, and 2 low severity issues.

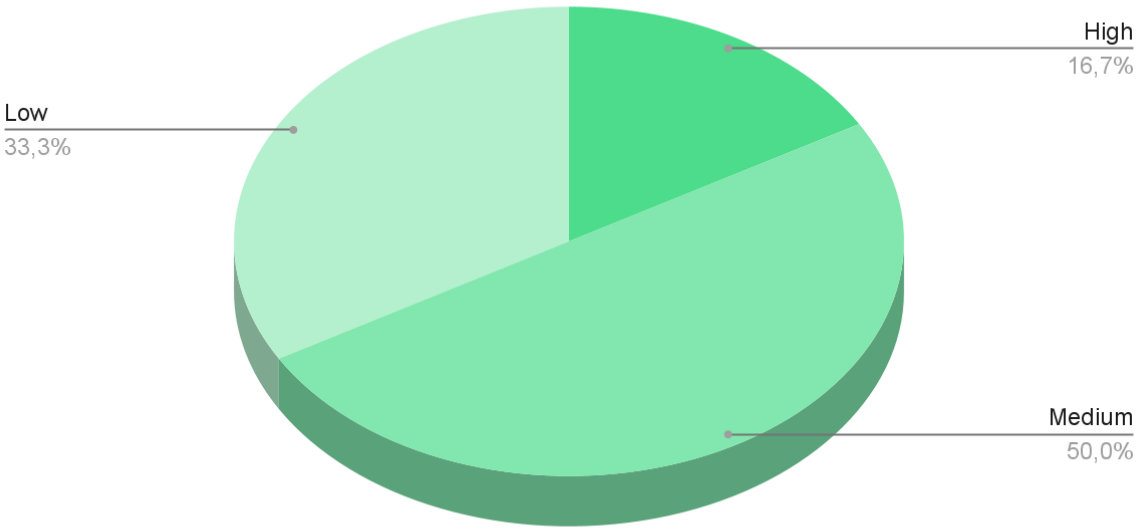
As a result of the second review, the Customers` smart contract contains 1 critical and 2 low severity issues.

As a result of the third review, the Customers` smart contract contains 2 low severity issues.

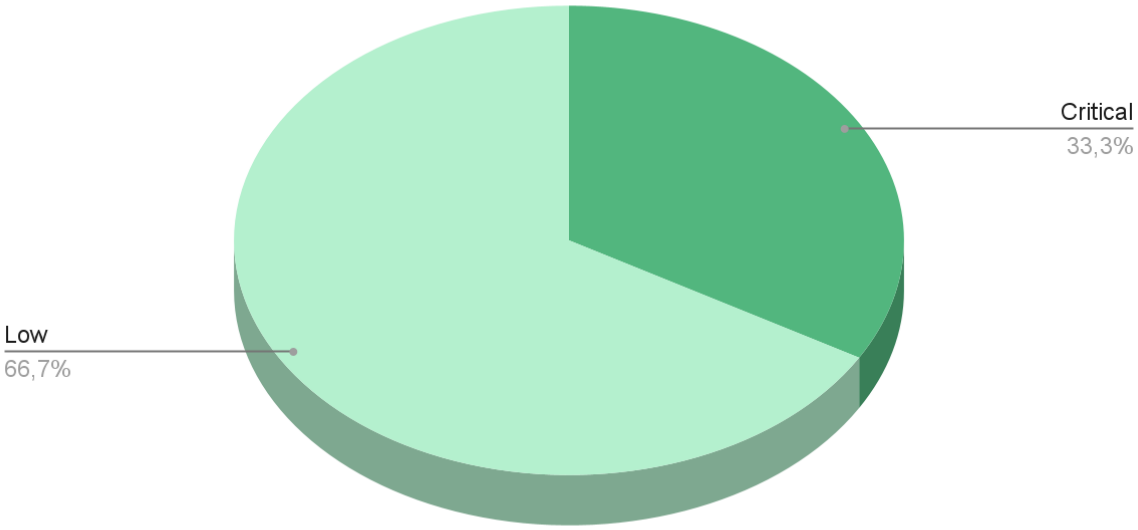
As a result of the fourth review, the Customers` smart contract contains no issues.

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

Graph 1. The distribution of vulnerabilities after the audit.

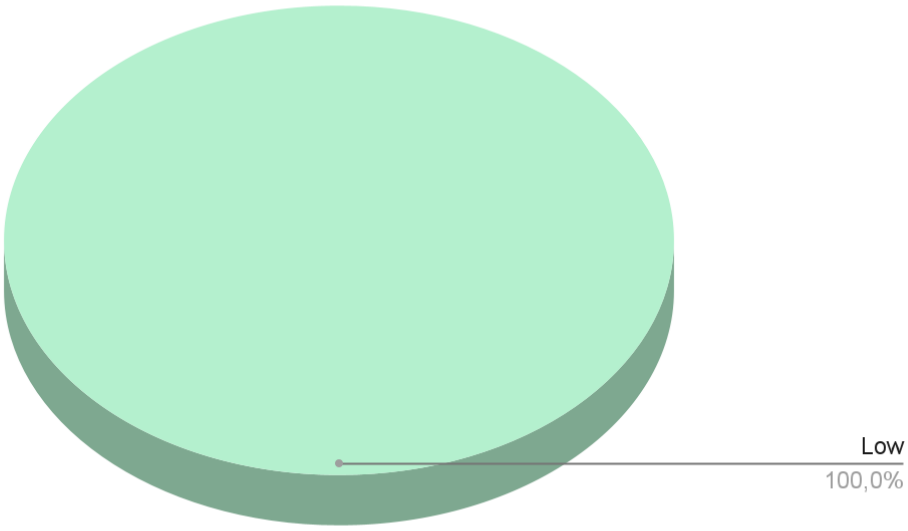


Graph 2. The distribution of vulnerabilities after the second review.



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

Graph 3. The distribution of vulnerabilities after the third review.



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

It's highly likely user will not get anything on `emergencyWithdraw` called. Due to zeroing `user.amount` and consequent transfering zero value to user with `safeTransfer` in the end - every user counter is zero and nothing transfered.

Contracts: KrownAutoCompoundFarm.sol

Status: fixed.

High

Rewards and deposit balances are not separated. If the contract is not filled by reward tokens, some users will not be able to withdraw their deposits after other users' rewards will be paid.

Contract: KrownAutoCompoundFarm.sol

Recommendation: split deposit and rewards balances so that all users will be able to withdraw all their tokens.

Status: fixed.

Medium

1. The `dailyBlocks` variable could be changed via `changeDailyBlocks`. In the meantime, variable `dailyBlocks` is used as divisor in `updateRewardsPerBlock`. There are no `require` call in `changeDailyBlocks`, so dailyBlocks could be `0`. This leads to division by zero error.

   Contract: KrownAutoCompoundFarm.sol

   Recommendation: add `require` function call for validating dailyBlocks argument inside `changeDailyBlocks` function.

   Status: fixed.

2. Library `SafeMath` is imported for `uint` but never used.

   Contract: KrownAutoCompoundFarm.sol

   Recommendation: remove library `SafeMath` from imports.

   Status: fixed.

3. The argument `_targetDailyCompoundRate` in changeTargetDailyCompoundRate required to be gretater than 1e18, but in `constructor` it could be anything.

   Contract: KrownAutoCompoundFarm.sol

   Recommendation: use a `require` function call to validate value of `_targetDailyCompoundRate`.

   Status: fixed.

Low

1. Limit value for fee of 10000 (100%) present as magic number in code.better to keep in constant.

   Contracts: KrownAutoCompoundFarm.sol

   Recommendation: it would be better to use a constant instead.

   Status: fixed.

2. Unnecessary calculation - the value `totalAmount` is calculated even if `user.divisor` is zero and zero result will be returned by `getPending` function.

   Contracts: KrownAutoCompoundFarm.sol

   Recommendation: order of calls in `getPending` function could be improved with early return statement.

   Status: fixed.

3. Duplicated logic with substracting fee in functions `withdraw` and `claim` - can be in separate method (for example - chargeFee).

   Contracts: KrownAutoCompoundFarm.sol

   Status: fixed.

4. Call of `require` in `claim` function could be made before substracting `pendingAmount` from `totalAmount`. Just right after pendingAmount is calculated.

   Contracts: KrownAutoCompoundFarm.sol

   Status: fixed.

5. Inside `deposit` function “magic number” 10000 still used as upper bound for fee instead of `MAX_FEE` constant.

   Contracts: KrownAutoCompoundFarm.sol

   Status: fixed.

6. Inside deposit function there is duplicated logic of calculating fee and transfering it to vault. It could be done using `chargeRemovalFee` function. Consider passing fee as second parameter. This issue relates to issue #3 of this list.

   Contracts: KrownAutoCompoundFarm.sol

   Status: fixed.

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 1 high, 3 medium, and 2 low severity issues.

As a result of the second review, the Customers` smart contract contains 1 critical and 2 low severity issues.

As a result of the third review, the Customers` smart contract contains 2 low severity issues.

As a result of the fourth review, the Customers` smart contract contains no issues.

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 the 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 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.