HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: RedFox
Date: September 30th, 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 RedFox.
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/RFL-Valt/smart-cheft-contract
Commit	dddf820a166b092fac4d1ad570fd2a424f6381d8
Technical Documentation	NO
Deployed contract	https://etherscan.io/address/0x4394f7d0b05f80baf246f79854e0e93f71181df1
JS tests	NO
Timeline	15 SEPTEMBER 2021 - 30 SEPTEMBER 2021
Changelog	20 SEPTEMBER 2021 - INITIAL AUDIT 22 SEPTEMBER 2021 - SECOND REVIEW 30 SEPTEMBER 2021 - THIRD 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 RedFox (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 15th, 2021 - September 20th, 2021. The second code review conducted on September 22nd, 2021. The third code review conducted on September 30th, 2021.

Scope

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

Repository:
https://github.com/RFL-Valt/smart-cheft-contract
Commit:
dddf820a166b092fac4d1ad570fd2a424f6381d8
Deployed contract:
https://etherscan.io/address/0x4394f7d0b05f80baf246f79854e0e93f71181df1
Technical Documentation: No
JS tests: No
Contracts:
interfaces\IERC20.sol
libraries\Address.sol
libraries\Context.sol
libraries\Ownable.sol
libraries\ReentrancyGuard.sol
libraries\SafeERC20.sol
libraries\SafeMath.sol
mock\ERC20.sol
SmartChefFactory.sol

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

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

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

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 **4** medium and **1** low severity issue.

After the second review security engineers found that all previously found medium severity issues were fixed however **1** low severity issue is still there.

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

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

1. Tautology or contradiction

   While IERC20.totalSupply() function returns **uint256**, which is an **unsigned** integer, the returned value will always be greater than or equal to zero. Therefore the expressions below would always return true and are excess.

   Recommendation: Remove excess “require” statements.

   Fixed before the second review

2. The pool owner is able to withdraw rewards

   Using declared function emergencyRewardWithdraw the pool owner/creator cold withdraw the entire rewards balance at any time.

   Recommendation: please either add some delay to let users take out their rewards in the case of some pool creator wants to scam, or remove that functionality.

   Fixed before the second review

3. The pool owner could stop rewards with no event emitted

   Using declared function stopReward the pool owner/creator cold stop rewards at any time with no event or notification for users.

   Recommendation: please at least emit a NewStartAndEndBlocks event on changing **bonusEndBlock**.

   Fixed before the second review

4. Some users could receive less or more than deserved

   By calling the updateRewardPerBlock the pool creator/owner could set a new value (even 0) and it will affect all previously staked but not withdrawn users.

   Recommendation: Please revise the pool rewards logic so you can recalculate previously earned rewards before changing the coefficient.

   Fixed before the second review

Low

1. A State variable that could be declared immutable

State variables that got initialized in the constructor and then never change their values should be declared immutable to save gas.

Recommendation: Please add an immutable attribute for state variables that are initialized in the constructor and never change their values.

Fixed before the third review

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.

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

As a result of the audit, security engineers found **4** medium and **1** low severity issue.

After the second review security engineers found that all previously found medium severity issues were fixed however **1** low severity issue is still there.

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

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, 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.