HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: FantOHM

Date: February 15th, 2022

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 FantOHM.
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/chinu-dev/fantohm-contract
Commit	DEE3179F67701D4CDC47709C3567D69CC418258D
Technical Documentation	Yes
JS tests	No
Website	https://app.fantohm.com/#/stake
Timeline	25 January 2022 - 15 February 2022
Changelog	02 February 2022 – Initial Audit 15 February 2022 – Second Audit

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by FantOHM (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 January 25th, 2022 - February 02nd, 2022.

The second review was conducted on February 15th, 2022.

Scope

The scope of the project is deployed smart contract:

Repository:
https://github.com/chinu-dev/fantohm-contract
Commit:
dee3179f67701d4cdc47709c3567d69cc418258d
Technical Documentation: Yes
JS tests: No
Contracts:
./contracts/stakingStaking/StakingStaking.sol
./contracts/stakingStaking/RewardsHolder.sol

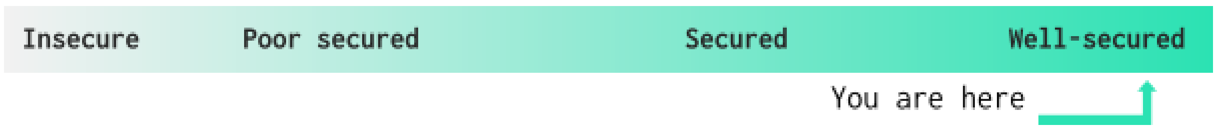
We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

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	▪ 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.



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** critical, **1** medium, and **1** low severity issues.

As a result of the second audit, security engineers found **no** issues. All previously reported issues have been fixed.

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

1. In either case, users should be able to withdraw their staked tokens. But there are several cases in the StakingStaking contract when users cannot withdraw funds:
   - Owners can enable whitelist and all non-whitelisted users will not be able to withdraw funds;
   - The pool can be empty until the last user exits;
   - emergencyWithdraw can be disabled by owners.

Contracts: StakingStaking.sol

Recommendation: make sure users can always withdraw their staked tokens.

Status: fixed.

2. Owners can destroy the contract with users' funds.

Contracts: StakingStaking.sol

Function: destroy

Recommendation: make sure all funds are withdrawn from the contract before destroying it.

Status: fixed.

■■■ High

No high severity issues were found.

■■ Medium

_amount can be greater than info.borrowed, so info.borrowed should be subtracted from totalBorrowed in this case, not _amount.

Contracts: StakingStaking.sol

Function: returnBorrow

Recommendation: fix it.

Status: fixed.

Low

The _shares variable is not used.

Contracts: StakingStaking.sol

Functions: deposit, withdraw, transfer

Recommendation: remove unused variables.

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 **2** critical, **1** medium, and **1** low severity issues.

As a result of the second audit, security engineers found **no** issues. All previously reported issues have been fixed.

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.