HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Minimax Finance
Date: February 16th, 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 Minimax Finance.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Deposits manager; BEP20 token; Governance; Staking; Vesting;
Platform	BSC / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/minimaxdefi/contracts/
Commit	1802DA3C6598CC46157C951B7789535C4C1F7B7C
Technical Documentation	YES
JS tests	YES
Website	https://www.minimax.finance/
Timeline	21 JANUARY 2022 - 15 FEBRUARY 2022
Changelog	27 JANUARY 2022 - INITIAL AUDIT 11 FEBRUARY 2022 - SECOND AUDIT 16 FEBRUARY 2022 - THIRD AUDIT

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

Table of contents

Introduction 4

Scope 4

Executive Summary 5

Severity Definitions 9

Audit overview 10

Conclusion 13

Disclaimers 14

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

Introduction

Hacken OÜ (Consultant) was contracted by Minimax Finance (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 21st, 2022 - January 27th, 2022.

The second review was conducted on February 11th, 2022.

The third review was conducted on February 16th, 2022.

Scope

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

Repository:
https://github.com/minimaxdefi/contracts/
Commit:
1802da3c6598cc46157c951b7789535c4c1f7b7c
Technical Documentation: Yes
JS tests: Yes
Contracts:
./contracts/helpers/BEP20.sol
./contracts/helpers/SafeBEP20.sol
./contracts/interfaces/IBEP20.sol
./contracts/interfaces/IMasterChef.sol
./contracts/interfaces/IMinimaxMain.sol
./contracts/interfaces/IMinimaxToken.sol
./contracts/interfaces/IPancakeRouter.sol
./contracts/interfaces/IPriceOracle.sol
./contracts/interfaces/ISmartChefInitializable.sol
./contracts/Migrations.sol
./contracts/MimimaxVesting.sol
./contracts/MinimaxMain.sol
./contracts/MinimaxStaking.sol
./contracts/MinimaxToken.sol
./contracts/ProxyCaller.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

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** critical, **2** high, **4** medium, and **2** low severity issues.

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

As a result of the second audit, security engineers found **1** new critical issue. Status of previously reported issues: **1** critical, **2** high, **2** medium, and **2** low severity issues were resolved, **2** medium issues remained.

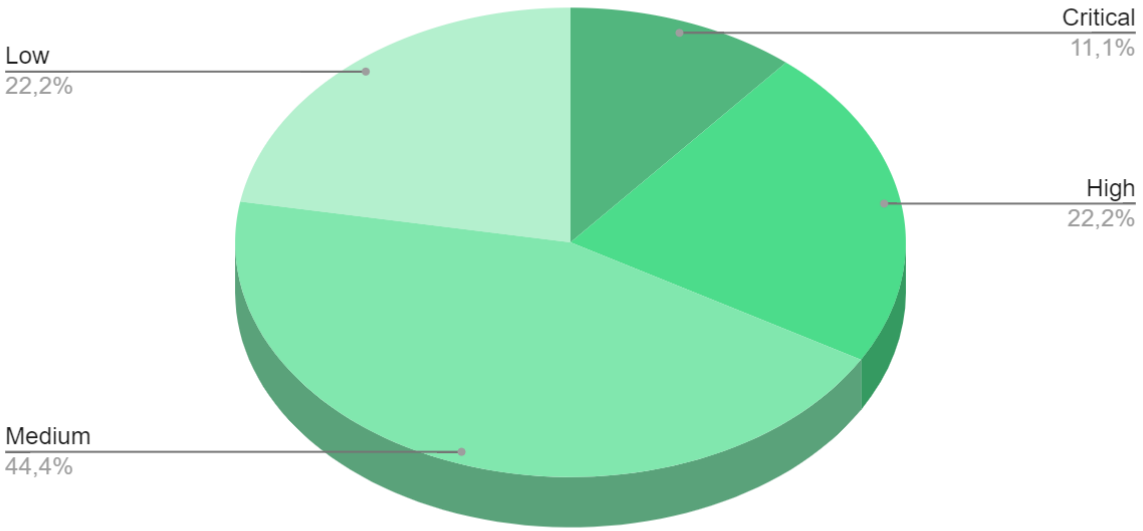
As a result of the third audit, security engineers found **no** new issues, **2** medium issues remained.

Notices:

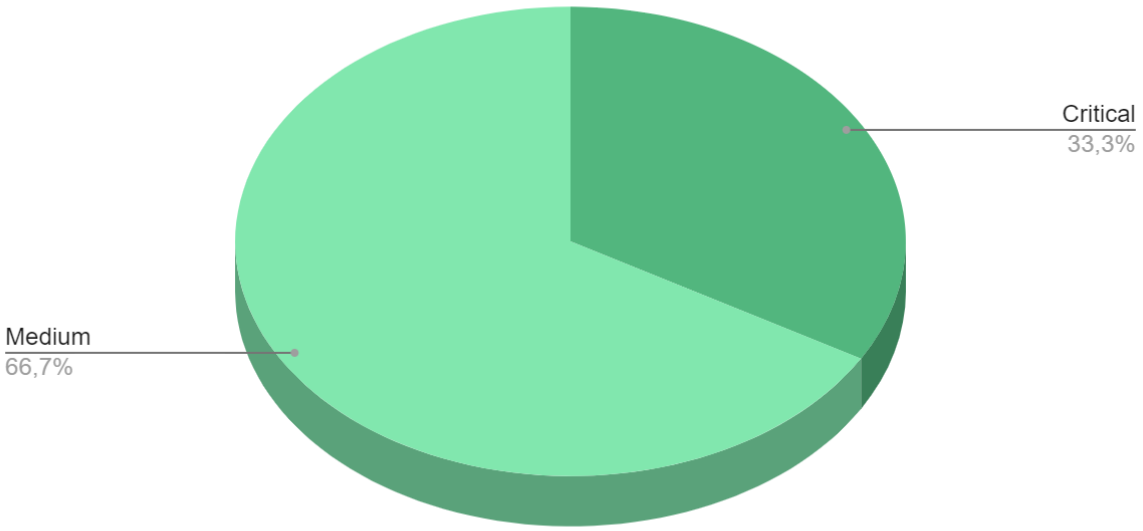
1. In the MinimaxStaking.sol contract, the pool parameters can be changed by the owners at any time. In addition, if the maximum MMX supply is reached, users will not be able to receive any staking rewards and can only withdraw their tokens.
2. Liquidation decision logic is executed offchain and is out of the audit scope. We may not guarantee its correctness.

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

Graph 1. The distribution of vulnerabilities after the audit.

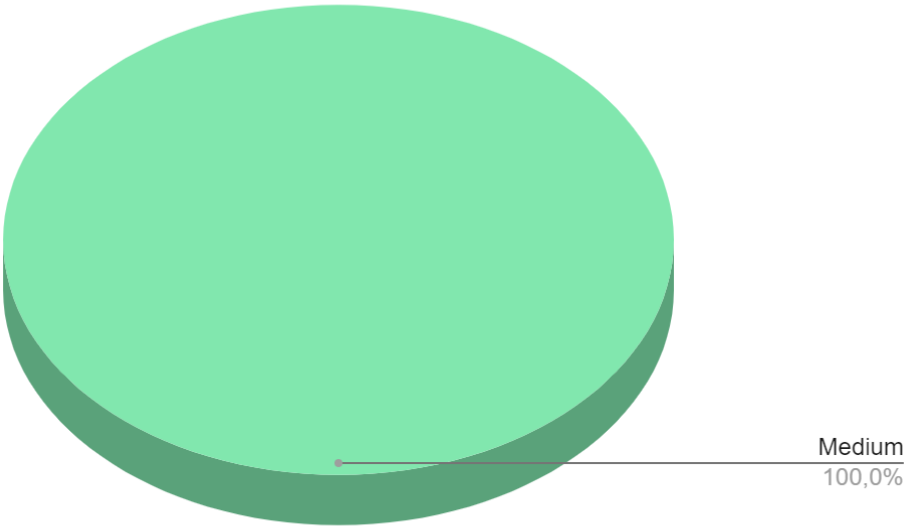


Graph 2. The distribution of vulnerabilities after the second audit.



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

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



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

1. The mint function of the BEP20 contract allows owners to mint tokens without restrictions.

Contracts: BEP20.sol

Function: mint

Recommendation: remove the possibility of unlimited minting.

Status: fixed.

2. If userFeeAmount is 0, then the amountToStake will be 0 and the user will lose all tokens that were transferred during the transaction.

Contracts: MinimaxMain.sol

Function: stakeCake

Recommendation: forbid stakes if userFeeAmount is 0.

Status: fixed.

High

1. According to the original design, the number of votes should be equal to the balance of tokens. In the code of this contract, this dependency is broken and the distribution of votes does not work correctly.

Contracts: MinimaxToken.sol

Recommendation: fix it.

Status: fixed.

2. Reward token address is not validated. Any address and any token could be passed.

Contracts: MinimaxMain.sol

Functions: stakeCake

Recommendation: fetch reward tokens from a pool.

Status: fixed.

Medium

1. Zero deposit allowed. This can lead to the loss of user funds for gas.

Contracts: MinimaxStaking.sol

Function: deposit

Recommendation: add a check to make sure the deposit amount is greater than 0.

Status: fixed.

2. May run out of gas if the amount parameter is too big.

Contracts: MinimaxMain.sol

Function: addNewCallers

Recommendation: limit the amount.

Status: not fixed.

3. Might run out of gas if positionIndexes array length is too big.

Contracts: MinimaxMain.sol

Function: liquidateManyByIndex

Recommendation: limit positionIndexes array length.

Status: not fixed.

4. Cake tokes are transferred to the MinimaxMain contract in the withdrawViaCaller function and transferred to a user in the withdrawImpl function. As a result, gas is paid for 2 transfers.

Contracts: MinimaxMain.sol

Function: withdrawImpl

Recommendation: send tokens directly to a user in the withdrawViaCaller function.

Status: fixed.

Low

1. _maxAmount is not checked before setting and can be larger than allowed for minting according to MAX_SUPPLY.

Contracts: MinimaxToken.sol

Function: setMinter

Recommendation: add a check.

Status: fixed.

2. The updatePool function has a redundant check because the previous checks in this function ensure that this check will always pass.

Contracts: MinimaxStaking.sol

Function: updatePool

Recommendation: remove redundant check.

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.

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

As a result of the audit, security engineers found **1** critical, **2** high, **4** medium, and **2** low severity issues.

As a result of the second audit, security engineers found **1** new critical issue. Status of previously reported issues: **1** critical, **2** high, **2** medium, and **2** low severity issues were resolved, **2** medium issues remained.

As a result of the third audit, security engineers found **no** new issues, **2** medium issues remained.

Notices:

3. In the MinimaxStaking.sol contract, the pool parameters can be changed by the owners at any time. In addition, if the maximum MMX supply is reached, users will not be able to receive any staking rewards and can only withdraw their tokens.
1. Liquidation decision logic is executed offchain and is out of the audit scope. We may not guarantee its correctness.

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.