HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Minterest
Date: April 13th, 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 Minterest.
Approved By	Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type of Contracts	Lending protocol
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://minterest.com/
Timeline	03.03.2022 - 13.04.2022
Changelog	29.03.2022 - Initial Review 13.04.2022 - Revising

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 Minterest (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 contracts.

Scope

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

Repository:
https://github.com/minterest-finance/protocol
Commit:
5784226e127f7c5903731d239761b5181a15f0bb
Technical Documentation:
- ReadMe File
- Comments in the code;

Functional Requirements:
- https://minterest.gitbook.io/sec-audit-doc/
- https://minterest.com/whitepaper

JS tests: Yes
Contracts:
/Governance/Mnt.sol
/Governance/MntGovernor.sol
/Governance/MntTimelock.sol
/Governance/MntVotes.sol
/Oracles/AggregatorV3Interface.sol
/Oracles/ChainlinkPriceOracle.sol
/Oracles/PriceOracle.sol
/Buyback.sol
/BuybackDripper.sol
/DeadDrop.sol
/EmissionBooster.sol
/EmissionCommission.sol
/ErrorCodes.sol
/InterestRateModel.sol
/KinkMultiplierModel.sol
/LinearRateModel.sol
/Liquidation.sol
/MinterestNFT.sol
/MintProxy.sol
/MNTSource.sol
/MToken.sol
/MTokenInterfaces.sol
/Supervisor.sol
/SupervisorInterface.sol
/SupervisorStorage.sol
/Treasury.sol
/Vesting.sol
/Whitelist.sol
/WhitelistInterface.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 ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ EIP standards violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency ▪ Kill-Switch Mechanism

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

Executive Summary

The score measurements details can be found in the corresponding section of the methodology.

Documentation quality

The Customer provided comprehensive functional requirements and technical requirements. The total Documentation Quality score is **10** out of **10**.

Code quality

The total CodeQuality score is **10** out of **10**. Code follows best practices. Test coverage is high.

Architecture quality

The architecture quality score is **10** out of **10**. All contracts are well-defined and well-structured. Cross-contract interactions are justified and do not cause tight-coupling of contracts.

Security score

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

As a result of the second review, security engineers found no new issues. All high, 3 medium, and all low severity issues were fixed. The code contains **1** medium severity issue. The security score is **10** out of **10**.

All found issues are displayed in the “Issues overview” section.

Summary

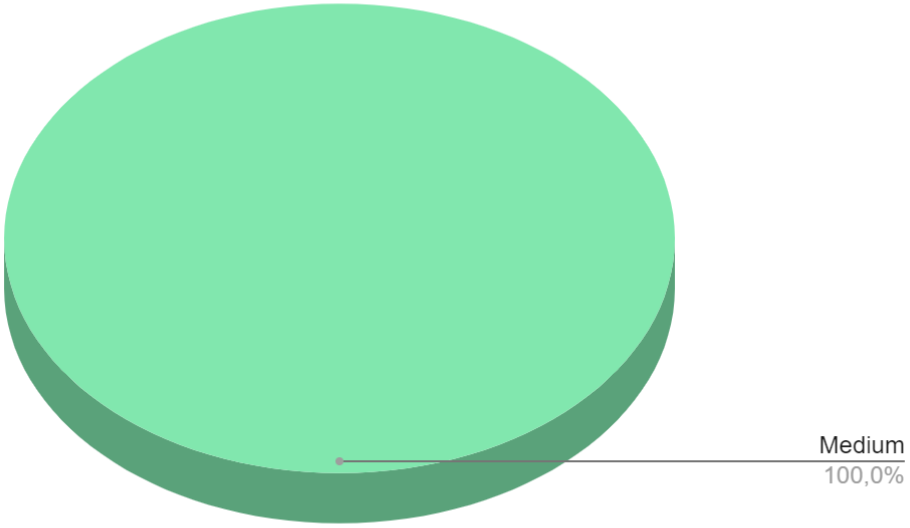
According to the assessment, the Customer's smart contract has the following score: **10.0**

Notices

1. All staking operations can be stopped by owners.
2. Vesting schedules can be revoked by owners if the corresponding “revocable” flag is set. The flag is set during schedule creation and can not be changed afterward.
3. The latest version of the code contains changes out of the second review scope. We cannot validate those changes.

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

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 cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution

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

Findings

Critical

No critical issues were found.

High

1. **Insufficient vesting balance.**

   The code validates that there are enough tokens only for every single vesting. However, there is no validation verifying that the contract balance is enough to fulfill all those vesting records.

   The contract does not guarantee that all users will receive their funds.

   Contracts: Vesting.sol

   Function: createVestingScheduleBatch

   Recommendation: implement an accounting of tokens transferred to the contract for vesting purposes and decrease this value with each new scheduled vesting.

   Status: Fixed (Revision: 15fcf36)

2. **Unrestricted function access.**

   The function can be called by everyone.

   This can lead to an undesired contract state.

   Contracts: EmmisionBooster.sol

   Function: updateBorrowIndexesHistory

   Recommendation: restrict access to the function.

   Status: Fixed (Revision: 15fcf36)

Medium

1. **Missing events emitting.**

   “MemberAdded” events are not emitted in the constructor when new addresses are whitelisted.

   If there is some off-chain logic that depends on “MemberAdded” event, it can work incorrectly.

   Contracts: Whitelist.sol

   Function: constructor

   Recommendation: Move whitelisting out of the constructor and make a function that accepts multiple accounts for whitelist.

   Status: Fixed (Revision: 15fcf36)

2. **Redundant modifiers.**

   The contract has redundant nonReentrant modifiers. As soon as no external calls are performed, nonReentrant modifier is redundant.

   Contracts: MToken.sol

   Recommendation: Remove redundant modifiers.

   Status: Acknowledged

3. **TODO notices.**

   The code contains a lot of “TODO” notices. This can indicate that the code is not finalized.

   Recommendation: Resolve all TODO notices.

   Status: Fixed (Revision: 15fcf36)

4. **Costly loops.**

   The code does not allow to enable emission boost for different markets in batches and process everything in a single call.

   The function can fail if the number of the markets returned by “supervisor.getAllMarkets()” is big enough.

   Contracts: EmmisionBooster.sol

   Function: enableTiersInternal

   Recommendation: re-think the function logic to avoid possible gas limit issues.

   Status: Mitigated

Low

1. **The contract can be declared as abstract.**

   The contract has some functions that should be implemented and never used separately.

   Contracts: SupervisorV1Storage.sol

   Recommendation: declare the contracts as abstract

   Status: Fixed (Revision: 15fcf36)

2. **Misleading naming**

   The function name says that it redeems something. However, it is a simple view function used for validation purposes.

   Contracts: Supervisor.sol

   Function: redeemAllowedInternal

   Recommendation: rename the function to display its purpose better

Status: Fixed (Revision: 15fcf36)

3. **Redundant addition**

   Adding 30 seconds for the current timestamp is redundant because the swap will be performed during the same call and using “block.timestamp” as the deadline is enough.

   Contracts: DeadDrop.sol

   Functions: swapTokensForExactTokens, swapExactTokensForTokens

   Recommendation: rename the function to display its purpose better

   Status: Fixed (Revision: 15fcf36)

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 by the best industry practices at the date of this report, with 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 a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements. 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 cannot guarantee the explicit security of the audited smart contracts.