HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: HODLVERSE

Date: January 24th, 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 HODLVERSE.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 token; Voting; IDO platform; Bridge;
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/hodlverse/contracts
Commit	94c670de90e97d6225847c19dd9d91ea97d1c9ce
Technical Documentation	YES
JS tests	YES
Website	https://hodlverse.com/
Timeline	01 DECEMBER 2021 - 24 JANUARY 2022
Changelog	04 DECEMBER 2021 - Initial Audit 10 DECEMBER 2021 - Second Review 24 JANUARY 2022 - 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 HODLVERSE (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 December 01st, 2021 - December 04th, 2021.

The second review was provided on December 10th, 2021.

The third review was provided on December 30th, 2021.

Scope

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

Repository:
https://github.com/hodlverse/contracts
Commit:
94c670de90e97d6225847c19dd9d91ea97d1c9ce
Technical Documentation: Yes
JS tests: Yes
Contracts:
./contracts/MoneyToken.sol
./contracts/IDO.sol
./contracts/BasicBridge.sol
./contracts/MainBridge.sol
./contracts/MoneyTokenBridge.sol
./contracts/SideBridge.sol
./contracts/interfaces/IERC20.sol
./contracts/interfaces/IBridgeToken.sol
./contracts/interfaces/IBEP20.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

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

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 **4** critical, **1** high and **2** medium severity issues.

As a result of the second review, the Customers` smart contracts contain **1** high severity issue.

As a result of the third review, all the issues are fixed.

Notice:

Bridge contracts are centralized and funds security depends on off-chain solutions used by the Customer to store and use private keys.

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 function is supposed to mint tokens to the recipient, but mints them to the Owner of the contract.

   Contracts: MoneyTokenBridge.sol

   Function: mintTo

   Status: Fixed

2. Contract owners can withdraw all tokens. Even tokens that have been sold but not yet claimed.

   Contracts: IDO.sol

   Function: withdrawRest

   Recommendation: remove the ability to withdraw tokens that have been sold.

   Status: Fixed

3. When the contract is suspended, users cannot claim their tokens.

   Contracts: IDO.sol

   Function: claim

   Recommendation: remove the whenNotPaused modifier or provide a contract that can be paused according to well-defined rules.

   Status: Fixed

4. Users can deposit tokens even if the contract does not have sufficient MONEY balance. No tokens will be received in the end of the sale in this case.

   Contracts: IDO.sol

   Function: deposit

   Recommendation: ensure that the contract MONEY balance is fulfilled before the sale.

   Status: Fixed

High

1. USDT and MONEY have different numbers of decimal places (6 and 18). This must be accounted for in exchangeRate. The exchangeRate value must pass more checks before being set.

   Contracts: IDO.sol

   Recommendation: add more validations for exchangeRate.

   Status: Fixed

2. Owners can withdraw sold but unclaimed tokens using the emergencyWithdraw function. If owners withdraw USDT before then, users who have not yet claimed will lose their funds.

   Contracts: IDO.sol

   Function: emergencyWithdraw

   Recommendation: users in any case must receive the purchased tokens or get their deposit back.

   Status: Fixed

Medium

1. If the approval amount exceeds the uint96 number range, it is better to set it to max uint96 than throw an error.

   Contracts: MoneyToken.sol, MoneyTokenBridge.sol

   Function: approve

   Recommendation: change the condition.

   Status: Fixed

2. Using the EXTCODESIZE check to prevent smart contracts from calling a function is not foolproof, it can be subverted by a constructor call, due to the fact that while the constructor is running, EXTCODESIZE for that address returns 0.

   Contracts: BasicBridge.sol

   Function: isContract

   Recommendation: Do not use the EXTCODESIZE check to prevent smart contracts from calling a function.

Status: Fixed

Low

No low severity issues were found.

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

As a result of the second review, the Customers` smart contracts contain **1** high severity issue.

As a result of the third review, all the issues are fixed.

Notice:

Bridge contracts are centralized and funds security depends on off-chain solutions used by the Customer to store and use private keys.

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.