HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Fintropy
Date: October 14th, 2021

This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed – upon a decision of the Customer.

Document

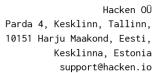|	
Name	Smart Contract Code Review and Security Analysis Report for Fintropy.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 tokens; Tokens Factory; Tokens Controller
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://gitlab.com/fintropyio/mvp-contracts
Commit	611944163831a8cea5faf02213b260ba3f7a8714
Technical Documentation	NO
JS tests	YES
Timeline	1 OCTOBER 2021 – 14 OCTOBER 2021
Changelog	11 OCTOBER 2021 – INITIAL AUDIT 14 OCTOBER 2021 – SECOND REVIEW

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Fintropy (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 October 1st, 2021 - October 14th, 2021.

Scope

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

Repository:
    https://gitlab.com/fintropyio/mvp-contracts
Commit:
    611944163831a8cea5faf02213b260ba3f7a8714
Technical Documentation: No
JS tests: Yes (https://gitlab.com/fintropyio/mvp-contracts/-/tree/master/test)
Contracts:
    PortController.sol
    PortFactory.sol
    PortToken.sol
    PortTokenUtils.sol
    VanilaERC20.sol
    WrappedToken.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.

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

After the second review security engineers found no security issues.

Notice:

Too many for-loops may cause contracts out-of-order.

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

No Critical severity issues were found

■■■ High

No high severity issues were found.

■■ Medium

1. External call in the loop

   Making an external call in the loop cause excess gas usage. It's more efficient to store the value to the local variable and access it in the loop instead.

   Contracts: PortController.sol
   Functions: issue, redeem
   Recommendation: please consider saving the value to the local variable.
   Status: Fixed.

2. Possible inaccessible contract

   View function which iterates through the array of undefined size could be out-of-work in the case when there would be a huge amount of values inside the array.

   Contracts: PortFactory.sol
   Functions: userPortTokens
   Recommendation: please consider changing the logic for calculating values without using for-loops.
   Status: Fixed by removing the function.

■ Low

1. A public function that could be declared external.

   public functions that are never called by the contract should be declared **external** to save gas.

   Contracts: PortController.sol, PortFactory.sol, WrappedToken.sol
   Functions: issue, redeem, issueFromOtherToken, issueFromWrappedToken, redeemToOtherToken, redeemToWrappedToken, createPortToken, userPortTokens, deposit, withdraw
   Recommendation: Use the **external** attribute for functions never called from the contract.
   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** medium and **2** low severity issues.

After the second review security engineers found no security issues.

Notice:

Too many for-loops may cause contracts out-of-order.

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.