HACKEN

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer**: Unore
**Date**: September 16th, 2021

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 Unore. |
| **Approved by** | Andrew Matiukhin \| CTO Hacken OU |
| **Type** | ERC20 token |
| **Platform** | Ethereum / Solidity |
| **Methods** | Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review |
| **Repository** | https://github.com/Uno-Re/actuary-group-smartcontract |
| **Commit** | 759bf927a581dda251ed19773b9ce2401baf6d64 |
| **Technical Documentation** | No |
| **JS tests** | Yes |
| **Changelog** | 19 August 2021 - Initial Audit<br>16 September 2021 - remediations check |

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 Unore (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on August 19th, 2021.

Second review conducted on September 16th, 2021.

# Scope

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

**Repository:** https://github.com/Uno-Re/actuary-group-smartcontract
**Commit:** 759bf927a581dda251ed19773b9ce2401baf6d64
**Technical Documentation:** No
**JS tests:** Yes
**Contracts:**

```
contracts/Cohort.sol
contracts/UnoERC20.sol
contracts/Actuary.sol
contracts/PremiumPool.sol
contracts/RiskPool.sol
contracts/libraries/TransferHelper.sol
contracts/factories/CohortFactory.sol
contracts/factories/RiskPoolFactory.sol
contracts/interfaces/IPremiumPool.sol
contracts/interfaces/IUnoERC20.sol
contracts/ClaimAssessor.sol
contracts/interfaces/ICohortFactory.sol
contracts/factories/PremiumPoolFactory.sol
contracts/interfaces/IRiskPoolFactory.sol
contracts/interfaces/IRiskPool.sol
contracts/Mocks/MockUSDT.sol
contracts/interfaces/ICohort.sol
contracts/interfaces/IPremiumPoolFactory.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:

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

| Category | Check Item |
| --- | --- |
| Code review | ▪ Reentrancy<br>▪ Ownership Takeover<br>▪ Timestamp Dependence<br>▪ Gas Limit and Loops<br>▪ DoS with (Unexpected) Throw<br>▪ DoS with Block Gas Limit<br>▪ Transaction-Ordering Dependence<br>▪ Style guide violation<br>▪ Costly Loop<br>▪ ERC20 API violation<br>▪ Unchecked external call<br>▪ Unchecked math<br>▪ Unsafe type inference<br>▪ Implicit visibility level<br>▪ Deployment Consistency<br>▪ Repository Consistency<br>▪ Data Consistency |
| Functional review | ▪ Business Logics Review<br>▪ Functionality Checks<br>▪ Access Control & Authorization<br>▪ Escrow manipulation<br>▪ Token Supply manipulation<br>▪ Assets integrity<br>▪ User Balances manipulation<br>▪ Data Consistency manipulation<br>▪ Kill-Switch Mechanism<br>▪ 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 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.

Recommendations regarding the project architecture or code structures can be found in the Recommendations section of the report.

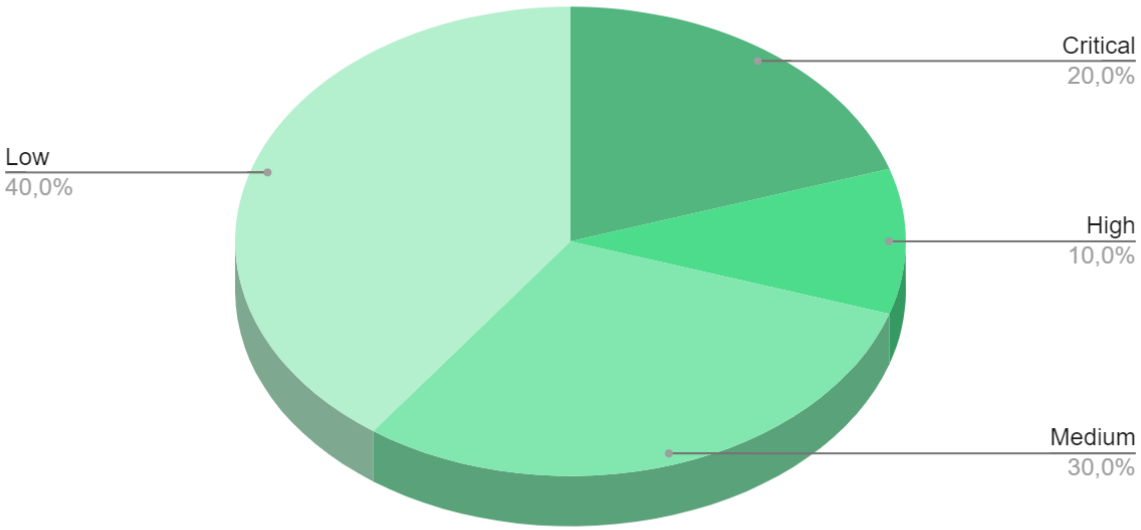
As a result of the audit, security engineers found **2** critical, **1** high, **3** medium, and **4** low severity issues.

As a result of the second review, Customers` smart contracts contain no issues.

**Notice:**

No technical and functional description is provided by the Customer. We may not guarantee correctness of calculations used in the code.

*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 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 contract owner can withdraw all funds from premium pools.

   **Contracts:** Cohort.sol

   **Function**: transferPremium

   **Recommendation:** contract owner should not be able to withdraw funds that does not belong to him.

   **Status:** fixed.

## High

1. The function can be called before `cohortActiveFrom` time and premium rewards will be locked on the low level.

   **Contracts:** Cohort.sol

   **Function:** leaveFromPool

   **Status:** fixed.

## Medium

2. New `Cohort` is supposed to be created via the `Actuary` contract but the `newCohort` function of the `CohortFactory` is not restricted from external access.

   **Contracts:** CohortFactory.sol

   **Function**: newCohort

   **Recommendation:** restrict access to the function.

   **Status:** fixed.

3. The function is unnecessarily declared as a `write` function. Though it can be simplified to a `read` function. `_premiumReward` value is set only once and never gets changed.

   **Contracts:** PremiumPool.sol

   **Function:** premiumRewardOf

   **Recommendation:** simplify the function.

   **Status:** fixed.

## Low

1. Comparison with `true` is redundant. The `isCohortCreator` function already returns a boolean value. And the requirement will pass if the function returns `true`.

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

**Contracts**: Actuary.sol

**Modifier:** onlyCohortCreator

**Recommendation**: remove redundant comparison.

**Status:** fixed.

2. The `_cohortAddr` variable is used only in the return statement. type(X).max

   **Contracts**: CohortFactory.sol

   **Function:** newCohort

   **Recommendation**: remove redundant local variable declaration.

   **Status:** fixed.

3. `MAX_INTEGER` value is hardcoded but solidity built in constant can be used instead.

   **Contracts**: Cohort.sol

   **Recommendation**: use type(uint256).max instead of hardcoded value.

   **Status:** fixed.

4. Result of the comparison can be set to `hasEnough` variable instead of explicitly set `true` or `false`.

   **Contracts**: Cohort.sol

   **Function:** hasEnoughCapital

   **Recommendation**: remove redundant comparison.

   **Status:** fixed.

## Recommendations

1. Initial pool capital is set as MAX_INTEGER without any need of such assignment. Such assignment leads to additional conditions in the `enterInPool`.

   **Contracts:** Cohort.sol

   **Function:** createRiskPool

   **Recommendation:** do not set a pool capital as MAX_INTEGER. The code can be simplified.

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.

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** high, **3** medium, and **4** low severity issues.

As a result of the second review, Customers` smart contracts contain no issues.

**Notice:**

No technical and functional description is provided by the Customer. We may not guarantee correctness of calculations used in the code.

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, bugfree 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.