HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Club Rare
Date: May 01st 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 Club Rare.
Approved By	Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type of Contracts	ERC20 token; Farming
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://doc.clubrare.com
Timeline	29.03.2022 - 01.05.2021
Changelog	01.04.2022 - Initial Review 15.04.2022 - Second Review 01.05.2022 - 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 Club Rare (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:
-
Commit:
-
Documentation: Yes - https://doc.clubrare.com/clubrare-universe/clubrare-utility-token-mpwr/clubrare-utility-token-mpwr-en
JS tests: No
Contracts:
AGOVToken.sol
MPWRToken.sol
ClubRareFarm.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 ▪ 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 functional requirements and no technical requirements. The total Documentation Quality score is **5** out of **10**.

Code quality

The total CodeQuality score is **5** out of **10**. Code follows official language style guides. No unit tests were provided.

Architecture quality

The architecture quality score is **10** out of **10**. Smart contracts of the project follow the best practices, and the project has a clear architecture.

Security score

As a result of the audit, security engineers found **6** medium and **5** low severity issues. The security score is **5** out of **10**.

After the second review, the code contains **3** medium and **2** low severity issues. The security score is **7** out of **10**.

After the third review, the code contains **1** medium severity issue. The security score is **10** out of **10**.

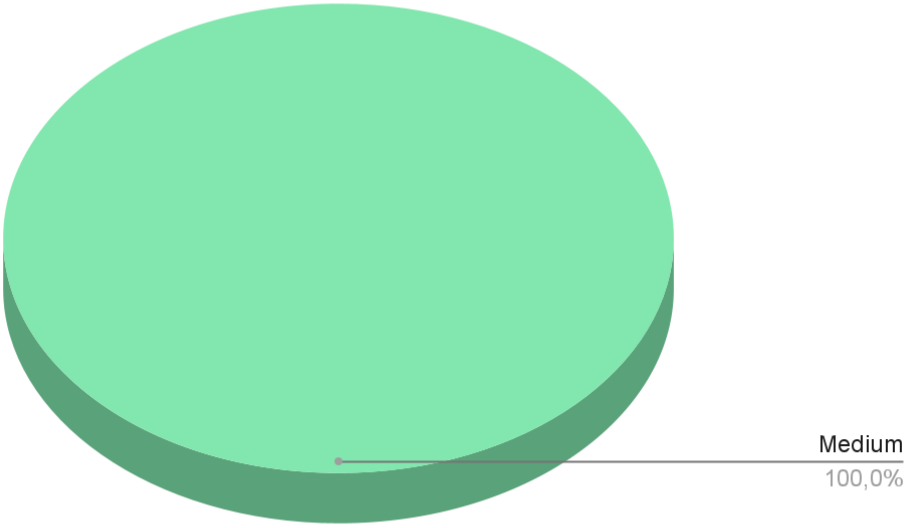
All found issues are displayed in the “Issues overview” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.0**

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 severity issues were found.

High

No high severity issues were found.

Medium

1. **Checks-Effects-Interactions Pattern Violation**

   Users' state variables are updated after MPWR token transfers from the farm contract.

   Contracts: ClubRareFarm.sol

   Function: withdraw()

   Recommendation: Send tokens out of the contract after user balances are decreased or zeroed.

   Status: Fixed

2. **Hardhat Console Deployed in Contract**

   "hardhat/console.sol" is in the deployment version of the contract.

   Contracts: ClubRareFarm.sol

   Function: -

   Recommendation: Remove *console.sol* imports and usages.

   Status: Fixed

3. **Unchecked Transfer**

   There are calls to ERC20's *transfer()* function, but their return values are never checked.

   This can lead to uneven behaviors in the function execution.

   Contracts: ClubRareFarm.sol

   Function: safeMpwrTransfer()

   Recommendation: Implement control mechanisms.

   Status: Fixed

4. **Missing _pid Existence Check.**

   Before interacting with the user-supplied poolInfo and userInfo parameters. Missing check whether _pid exists or not.

   Contracts: ClubRareFarm.sol

   Function: deposit()

   Recommendation: Implement control mechanisms.

   Status: Fixed

5. **Missing Allowance Check**

   The *deposit()* function assumes the caller has enough allowance to burn tokens.

   This could lead to reverts in the execution.

   Contracts: ClubRareFarm.sol

   Function: deposit()

   Recommendation: Implement control mechanisms for allowance checking.

   Status: Fixed

6. **Possible Out-of-Gas Exception**

   Iterating over pools may lead to enormous Gas consumption due to the pools' size.

   This could lead to a potential Out-of-Gas exception.

   Contracts: ClubRareFarm.sol

   Function: massUpdatePools()

   Recommendation: Review and check logic.

   Status: Reported

Low

1. **Missing Zero Address Validation.**

   Parameters of address type should be checked for potential zero addresses before use.

   Contracts: ClubRareFarm.sol, MPWRToken.sol

   Function: constructor(), mint(),

   Recommendation: Implement a zero address check.

   Status: Fixed

2. **Use of Hardcoded Value**.

   Hardcoded values are used in computations.

   Contracts: ClubRareFarm.sol

   Function: withdraw(), pendingMpwr()

   Recommendation: Move hardcoded values to constants.

   Status: Fixed

3. **Floating Pragma.**

   The project uses floating pragma ^0.8.0.

   Contracts: ClubRareFarm.sol, MPWRToken.sol, AGOVToken.sol

   Function: -

   Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

   Status: Fixed

4. **Outdated Solidity Version.**

   Using an old version prevents access to new Solidity security checks.

   Contracts: ClubRareFarm.sol, MPWRToken.sol, AGOVToken.sol

   Function: -

   Recommendation: Consider using one of these versions: *0.8.6, 0.8.9*.

   Status: Fixed

5. **Functions that can be Declared as *external*.**

   To save Gas, public functions that are never called in the contract should be declared as *external*.

   Contracts: MPWRToken.sol, AGOVToken.sol

   Function: decimals()

   Recommendation: Aforementioned function should be declared as *external*.

   Status: Mitigated. Overrides ERC20 implementation.

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.