HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Wonder Hero
Date: October 21st, 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 Wonder Hero.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 token
Platform	Polygon / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/devwonderhero/wonder-hero-contract-audit
Commit	39fc9472b5e236ee3877e221afef38ca82991051
Technical Documentation	YES
JS tests	NO
Website	https://www.wonderhero.io
Timeline	11 OCTOBER 2021 – 21 OCTOBER 2021
Changelog	21 OCTOBER 2021 – INITIAL AUDIT

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Wonder Hero (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 11ᵗʰ, 2021 - October 21ˢᵗ, 2021.

Scope

The scope of the project is smart contracts in the repository:
Repository:
    https://github.com/devwonderhero/wonder-hero-contract-audit
Commit:
    39fc9472b5e236ee3877e221afef38ca82991051
Technical Documentation: Yes (https://github.com/devwonderhero/wonder-hero-contract-audit/tree/master/puml)
JS tests: No
Contracts:
    HonorToken.sol
    WonderToken.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 secured.

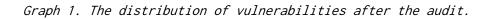| Insecure	Poor secured	Secured	Well-secured

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
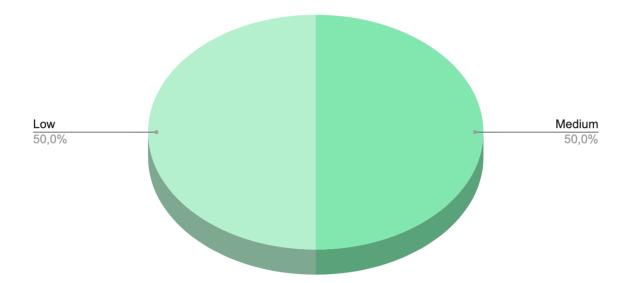
analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **1** medium and **1** low severity issues.

Graph 1. The distribution of vulnerabilities after the audit.

Low
50,0%

Medium
50,0%

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 issues were found.

■ ■ ■ High

No high severity issues were found.

■ ■ Medium

Centralization issue

The owner could add any address as the MINER_ROLE and it will be able to mint any amount of tokens anytime with no predefined cap.

Contracts: WonderToken.sol

Function: mint

Recommendation: Please make sure you've described in the docs the unlimited minting possibility to the owner of the contract. The other suggestion is to move the ownership of the contract to some Timelock to give the community time to react to unexpected minting issues.

■ Low

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: HonorToken.sol

Functions: addMiner, burn, burnFrom, mint, proxyBurn

Recommendation: Use the **external** attribute for functions never called from the contract.

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

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.