HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: HeroVerse
Date: November 8th, 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 HeroVerse.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 token; ERC721 token; Randomizer
Platform	Binance Smart Chain / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/Heroverse/core/tree/audit_final
Commit	a60da8a0cf7bef3dbc22df78ba2a7e7e3aa6edca
Technical Documentation	YES
JS tests	NO
Website	heroverse.io
Timeline	03 NOVEMBER 2021 - 08 NOVEMBER 2021
Changelog	08 NOVEMBER 2021 - INITIAL AUDIT

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 HeroVerse (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 November 3rd, 2021 - November 8th, 2021.

Scope

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

Repository:
https://github.com/Heroverse/core
Commit:
a60da8a0cf7bef3dbc22df78ba2a7e7e3aa6edca
Technical Documentation: Yes (provided in text)
JS tests: No
Contracts:
interfaces/IHeroNFT.sol
interfaces/ISharedStruct.sol
HeroNFT.sol
ChainlinkRandomNumberGenerator.sol
common/Operatable.sol
OraiRandomNumberGenerator.sol
interfaces/IVRFOracleOraichain.sol
interfaces/IRandomNumberGenerator.sol
HeroVerseToken.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 **6** low severity issues.

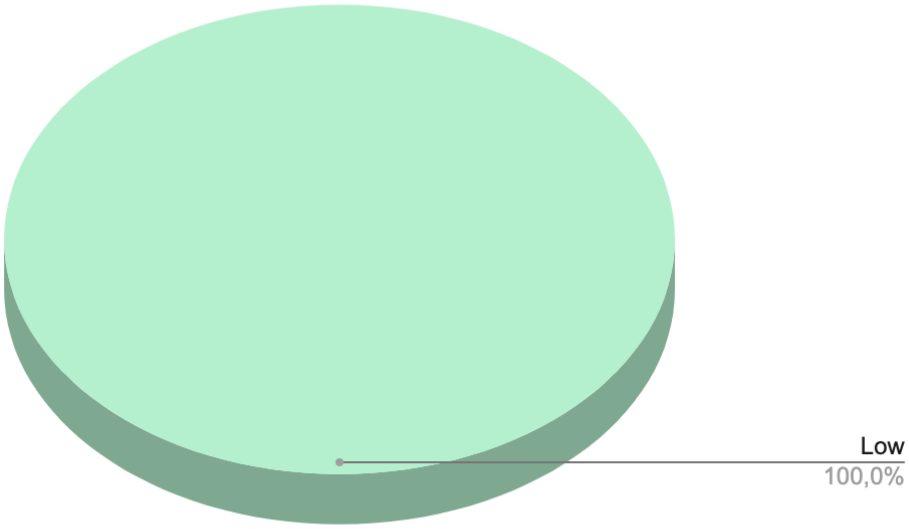
Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Notice:

According to the provided information from the customer, week-randomness functions would be used only in case of emergency when VRF is not working.

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 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

No critical issues were found.

High

No high severity issues were found.

Medium

No medium severity issues were found.

Low

1. Multiplication and power readability.

It's is recommended to use scientific notation and ether unit suffix instead of multiplication and power of ten. While the compiler will definitely transform it to the constant in compile-time, scientific notation with ether suffix could be more readable.

Contract: HeroVerseToken.sol

Function: constructor

Recommendation: Please try to use more efficient writing like "*1e9 ether*" instead.

2. Missing zero address validation.

Calling constructor it is possible to set mistaken treasury and heroVerseToken as zero addresses and then there is no possibility to modify those. That could lead to unexpected results.

Contract: HeroNFT.sol

Functions: constructor

Recommendation: Please verify that treasury and heroVerseToken are not zero-address.

3. State variables that could be declared immutable.

State variables that are initialized in the constructor and never change their values should be declared immutable to save gas.

Contract: HeroNFT.sol

Variable: treasury

Recommendation: Add the **immutable** attributes to state variables that never change and are initialized in the constructor.

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

4. Boolean equality.

Boolean constants can be used directly and do not need to be compared to **true** or **false**.

Contract: HeroNFT.sol

Function: _generateHero

Recommendation: Remove the equality to the boolean constant.

5. A public function that could be declared external.

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

Contract: ChainlinkRandomNumberGenerator.sol

Functions: withdrawTokens, requestRandomNumber

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

6. A public function that could be declared external.

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

Contract: OraiRandomNumberGenerator.sol

Functions: transferOnlyOwner, requestRandomNumber

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

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.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found **6** low severity issues.

Notice:

According to the provided information from the customer, week-randomness functions would be used only in case of emergency when VRF is not working.

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.