HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Wonder Hero
Date: December 24th, 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 Wonder Hero.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC721 tokens; Staking
Platform	Polygon / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/devwonderhero/wonder-hero-contract-audit
Commit	455354ec6aefa997aa6554c41631c9116691e414
Technical Documentation	NO
JS tests	NO
Website	wonderhero.io
Timeline	13 DECEMBER 2021 - 24 DECEMBER 2021
Changelog	24 DECEMBER 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 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 December 13th, 2021 - December 24th, 2021.

Scope

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

Repository:
https://github.com/devwonderhero/wonder-hero-contract-audit
Commit:
455354ec6aefa997aa6554c41631c9116691e414
Technical Documentation: No
JS tests: No
Contracts:
WonderAccount.sol
WonderMeta.sol
WonderOperator.sol
WonderOperatorETH.sol
access/WonderRoles.sol
box/WonderBoxClub.sol
box/BadgeBoxClub.sol
box/WonderBoxClubETH.sol
interface/IAccount.sol
interface/IWonderMarketPlace.sol
interface/IWonderOperatorETH.sol
interface/INFTStaking.sol
interface/IMarketLp.sol
interface/IFeeDistribution.sol
interface/IWonderBoxClub.sol
interface/IWonderMarketPlaceETH.sol
interface/IWonderOperator.sol
interface/IBoxNFT.sol
interface/IWonderBox.sol
interface/IWonderBoxClubETH.sol
interface/ISellAble.sol
interface/IWonderTicket.sol
interface/IWonderERC20.sol
interface/IMetaData.sol
interface/IWonderRoles.sol
interface/IGameNFT.sol
interface/IStakeAbleNFT.sol
interface/IBadge.sol
interface/IReferralShip.sol
libraries/WonderHeroSign.sol
marketplace/WonderMarketPlaceETH.sol
marketplace/WonderMarketPlace.sol
marketplace/MarketLP.sol
referralShip/FeeDistribution.sol
referralShip/ReferralShip.sol
staking/NFTStaking.sol
staking/NFTStakingLP.sol
staking/WonderTicket.sol

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

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

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 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.

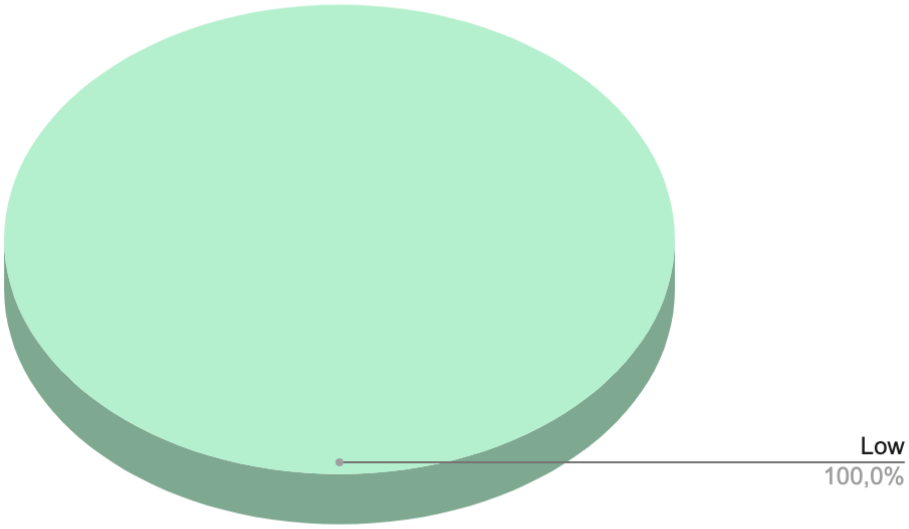
As a result of the audit, security engineers found **7** low severity issues.

Notice:

PRNG using in contract BadgeBoxClub.sol is vulnerable to attacks from malicious miners and should not be used to make decisions that could affect real assets. Even if a malicious user is not able to mine a block, it’s still possible to precalculate results and select a time slot that is more beneficial in terms of outcome. (Block time on Polygon chain is only ~2 seconds and is pretty stable, so there are plenty of options to select from)

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. The function iterates over or returns an array of unpredictable size

   Contracts: WonderRoles.sol

   Functions: getAddressList, transferGasToOperators, getBalanceList, getBalance

   Gas consumption grows with array size and starting from a certain size function could become inoperable.

   Recommendation: limit *_addressList[]* size

2. Redundant code

   Contracts: ReferralShip.sol

   Functions: getReferral

   Code block inside **if** statement is equal to code block outside of **if** statement

   Recommendation: change function

3. Boolean equality

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

   Contracts: BadgeBoxClub.sol, WonderBoxClub.sol, WonderMarketPlace.sol

   Functions: migrate, claimWonderBox, dealNFT

   Recommendation: remove the equality to the boolean constant.

4. State variables that could be declared constant

   Constant state variables should be declared constant to save gas.

   Contracts: WonderOperatorETH.sol, WonderBoxClubETH.sol, FeeDistribution.sol

   Variables: buyFee, productIdIndex, ONE, marketFeeRate

   Recommendation: Add the constant attributes to state variables that never change.

5. Missing event for changing *startTime*, *duration*, *_startTime*, *_perUserMaxCount*, *_duration*, *_roles*, *referralShip*, *startTime*, *finishTime*, *ticketPeriod*, *totalCount*, *ticketPrice*, *withdrawFee*, *marketPlaceEth*, *distributionAddr*, *marketPlace*, *wonderBoxContract*, *metaContract*, *_wonderRoles*

   Contracts: BadgeBoxClub.sol, WonderBoxClubETH.sol, WonderBoxClub.sol, FeeDistribution.sol, WonderTicket.sol, WonderOperatorETH.sol, WonderOperator.sol, NFTStaking.sol

   Functions: setPeriod, setBasic, setRoles, setReferralShip, setWithdrawFee, setMarketPlace, setFeeDistribution, setWonderBox, setWonderMeta, setAuth

   Changing critical values should be followed by the event emitting for better tracking off-chain.

   Recommendation: Please emit events on the critical values changing.

6. Using SafeMath in Solidity >= 0.8.0

   Starting solidity version 0.8.0 arithmetic operations revert on underflow and overflow. There’s no more need to assert the result of operations.

   Contracts: NFTStaking.sol

   Recommendation: Please avoid using assert for arithmetic operations.

7. 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: BadgeBoxClub.sol, FeeDistribution.sol, MarketLP.sol, NFTStaking.sol, NFTStakingLP.sol, ReferralShip.sol, WonderAccount.sol, WonderBoxClub.sol, WonderBoxClubETH.sol, WonderHeroSign.sol, WonderMarketPlace.sol, WonderMarketPlaceETH.sol, WonderMeta.sol, WonderOperator.sol, WonderOperatorETH.sol, WonderRoles.sol

Functions: addChainIds, addMeta, batchRegister, claimBadge, clearMeta, getEndTime, getRoles, getStartTime, setPeriod, calWinFee, init, burn, getLpInfo, initialize, mint, setLpInfo, getAuth, getBasicInfo, getSupportLpAddress, isSupportStake, burn, mint, setAuth, getReferral, getReferralFee, addItems, costItems, getAccount, getProperty, setAccountStatus, setDetail, toggleGlobalTransfer, upgradeAccount, claimWonderBox, claimFee, claimToken, deposit, getProduct, getStartTime, getUserLeft, getRoles, setRoles, getSignerV2, verifyPersonal, cancelNFT, dealNFT, buyNFTClient, claimFee, claimToken, dealNFTClient, getCurrency, setFeeDistribution, getSupportLpAddress, isSupport, ownerOf, sellNFT, addNftMeta, getChangeLevelCountByKind, getChangeLevelCount, getChangeMetaById, getChangeMetaByKind, getLevelCountByKind, getLevelCount, getLevelMetaById, getLevelMetaByKind, getNftMetaByIndex, getNftMeta, setChangeMeta, setLevelMeta, setNFTMeta, cancelNFTOperator, changeNFT, claimFee, claimToken, dealNFTOperator, depositOperator, depositClient, init, payWithdrawFee, withdrawClient, getOrderInfo, init, openWonderBox, sellNFTOperator, upgradeNFT, useItem, win, withdrawOperator, claimFee, claimGas

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 **7** low severity issues.

Notice:

PRNG using in contract BadgeBoxClub.sol is vulnerable to attacks from malicious miners and should not be used to make decisions that could affect real assets. Even if a malicious user is not able to mine a block, it's still possible to precalculate results and select a time slot that is more beneficial in terms of outcome. (Block time on Polygon chain is only ~2 seconds and is pretty stable, so there are plenty of options to select from)

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.