HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Valhalla
Date: March 25th, 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 Valhalla.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 token; Transfer controller
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/ValhallaLand/Solidity - - Initial Audit https://github.com/ValhallaLand/Solidity/commit/9a42f0d87132d1f39005f259b0a8b1ff92f56405 - Remediations check https://github.com/ValhallaLand/Solidity/tree/main/erc20 - Remediations Check v2
Commit	70de7d9daccb6d812b2ab96f20fb4aa2605e2601 - Initial Audit 9a42f0d87132d1f39005f259b0a8b1ff92f56405 - Remediations check 7af48f29c39a2b594726989a54478260ef3da6a1 - Remediations Check v2
Technical Documentation	No
JS tests	No
Website	https://valhalla.land
Timeline	14 FEBRUARY 2022 - 25 MARCH 2022
Changelog	16 FEBRUARY 2022 - Initial Audit 15 MARCH 2022 - Remediations check 25 MARCH 2022 - Remediations Check v2

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

Table of contents

Introduction 4

Scope 4

Executive Summary 5

Severity Definitions 7

Audit overview 8

Conclusion 8

Disclaimers 10

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

Introduction

Hacken OÜ (Consultant) was contracted by Valhalla (Client) 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 February 14th, 2022 - February 16th, 2022.

The second review was conducted on March 15th, 2022.

The third review was conducted on March 25th, 2022.

Scope

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

Repository:
https://github.com/ValhallaLand/Solidity - Initial Audit
https://github.com/ValhallaLand/Solidity/commit/9a42f0d87132d1f39005f259b0a8b1ff92f56405 - Remediations Check
https://github.com/ValhallaLand/Solidity/tree/main/erc20 - Remediations Check V2

Commit:
70de7d9daccb6d812b2ab96f20fb4aa2605e2601 - Initial Audit
9a42f0d87132d1f39005f259b0a8b1ff92f56405 - Remediations Check
7af48f29c39a2b594726989a54478260ef3da6a1 - Remediations Check V2

Technical Documentation: No

JS tests: No

Contracts:
Address.sol
Context.sol
IERC20.sol
IMyNFT.sol
Ownable.sol
ERC20.sol

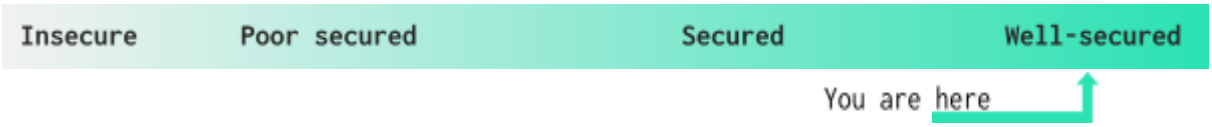
Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.



Our team analyzed code functionality, manual audit, and automated checks with Mythx 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** high, **1** medium, **7** low severity issues.

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

As a result of the remediations review, Customers' smart contracts contain **1** medium and **1** low severity issues.

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

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 asset 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 asset 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

Audit overview

Critical

No critical issues detected

High

The contract uses the same balance for liquidity and vestings.

Although there are separate balance variables for liquidity and vestings. They are stored together in the balances array. This leads to a common misconception that the vesting and liquidity balance are stored together in the _balances. Which eventually will allow a user to claim locked vesting or liquidity balance.

Contracts:ERC20.sol

Functions: -

Recommendation: Separately store them.

Status: Fixed

Medium

The ERC20 contract does not follow the single responsibility principle and implements more logic on top of the token. Overall system complexity is increased without visible benefits.

Contracts: ERC20.sol

Recommendation: separate vesting and IDO functionality from the token functionality.

Status: Fixed

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

Low

1. Missing Zero Address Validation.

In the constructor, the verifier is set to an address but never checked if it is a zero address.

In the addVestingLock function, the *_wallet* parameter is never checked if it's a zero address.

Contracts:ERC20.sol

Function: Constructor, addVestingLock(address _wallet,uint256 _amount, uint256 _deadline)

Lines: # 64 and between #208 - #216

Recommendation: Implement a missing zero address check.

Status: Fixed

2. Use of Hardcoded Values

In the constructor, initTreasure, and splitSignature functions, hardcoded values are used in calculations

Contracts: ERC20.sol

Functions: splitSignature(bytes memory sig), constructor, initTreasure()

Lines: between #12 - #183, between #229 - #246 and between #339 - #360

Recommendation: Move hardcoded values to constants.

Status: Fixed

3. Floating Pragma.

ERC20.sol uses floating pragma ^0.8.0.

Contracts: ERC20.sol

Function: -

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

Status: Fixed

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

4. State Variables that could be Declared as *constant*

State variables that don't change their (_decimals, _marketingWallet, _seedWallet, _teamWallet, _privateWallet) value should be declared constant to save gas.

Contracts:ERC20.sol

Functions: -

Recommendation:Declare above mentioned variables as constants.

Status: Fixed

5. Functions that can be Declared as *external*

In order to save gas, public functions that are never called in the contract should be declared as *external*.

Contracts: FeedLp.sol

Functions: name(), symbol(), decimals(), totalSupply(), balanceOf(address account), transfer(address recipient, uint256 amount), approve(address spender, uint256 amount), increaseAllowance(address spender, uint256 addedValue), decreaseAllowance(address spender, uint256 subtractedValue)

Recommendation: Aforementioned should be declared as *external*.

Status: Fixed

6. Incorrect Solidity Version

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

Contracts: ERC20.sol

Functions: -

Recommendation: Consider using one of these versions:*0.8.4, 0.8.6.*

Status: Fixed

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 **1** high, **1** medium, **7** low severity issues.

As a result of the remediations review, Customers' smart contracts contain **1** medium and **1** low severity issues.

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

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, about 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.