HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Golden Haven Games
Date: March 14th 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 Golden Haven Games.
Approved by	Andrew Matiukhin | CTO Hacken OU Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type	ERC20 token;
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/josemesa121/POA_TOKEN - Initial Audit https://github.com/josemesa121/POA_TOKEN/commit/d6dbaa4d817dcdf8a3d4f4dfe67d764fc2954b99 - Remediations Check
Commit	629B7BC9EE98E3CA3BC7948B33841A1A53419D91 - Initial Audit d6dbaa4d817dcdf8a3d4f4dfe67d764fc2954b99 - Remediations Check
Technical Documentation	YES - https://path-of-alchemist-1.gitbook.io/path-of-alchemist/
Website	https://path-of-alchemist-1.gitbook.io/path-of-alchemist/
Timeline	25 FEB 2022 - 14 MARCH 2022
Changelog	28 FEBRUARY 2022 - Initial Audit 14 MARCH 2022 - Remediations Check

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 Golden Haven Games (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:
https://github.com/josemesa121/POA_TOKEN - Initial Audit
https://github.com/josemesa121/POA_TOKEN/commit/d6dbaa4d817dcdf8a3d4f4dfe67d764fc2954b99 - Remediations Check
Commit:
629b7bc9ee98e3ca3bc7948b33841a1a53419d91 - Initial Audit
d6dbaa4d817dcdf8a3d4f4dfe67d764fc2954b99 - Remediations Check
Technical Documentation: Yes - https://path-of-alchemist-1.gitbook.io/path-of-alchemist/
JS tests: No
Contracts:
token_poa.sol
2_Owner.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

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

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

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 provided documents include information about tokenomics and fees. Total Documentation Quality score is **5** out of **10**.

Code quality

The total CodeQuality score is **5** out of **10**. No unit tests were provided. On the other hand, the coding guidelines are followed.

Architecture quality

The architecture quality score is **5** out of **10**. The logic inside the functions was too complex, and the contracts did not care about the Gas optimization, which led to high Gas costs.

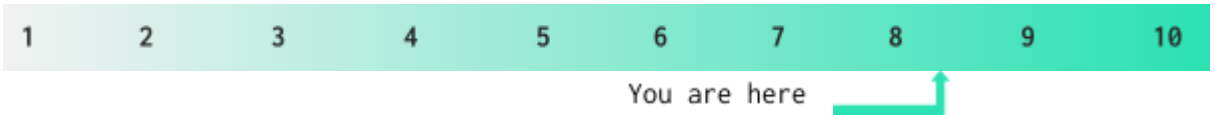
Security score

As a result of the audit, security engineers found **9** low severity issues. The security score is **7** out of **10**. All found issues are displayed in the “Issues overview” section.

As a result of the remediations review, Customers’ smart contracts contain **1** low severity issue. Therefore the score is updated to **10** out of **10.**

Summary

According to the assessment, the Customer's smart has the following score: **8.5**



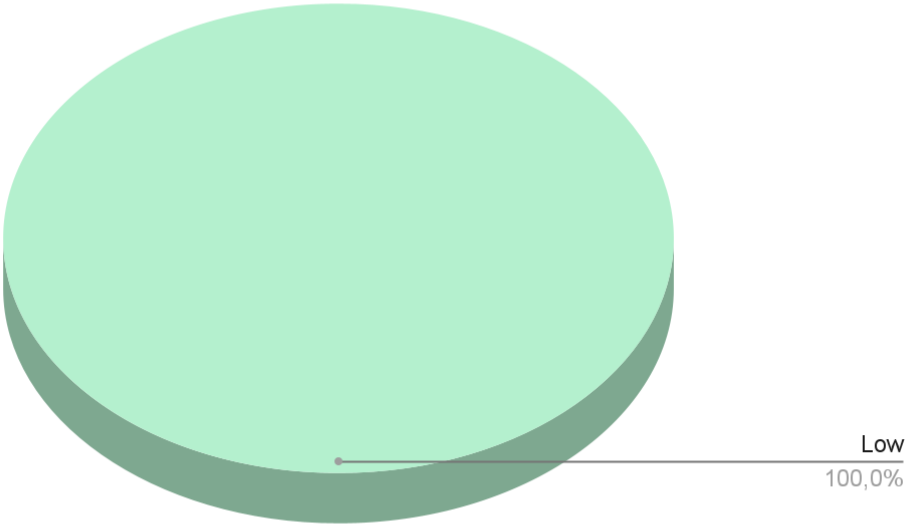
Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

NOTICE: The contract applies fees in the transfer functions (*transfer()* and *transferFrom()*). This leads a user to get fewer funds than the desired amount. Balance checks should be applied before and after each transfer to prevent any issues..

NOTICE: The logic in the *transfer()* and *transferFrom()* functions is too complex. This will lead users of this contract to pay large amounts of Gas fees in addition to business logic fees.

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

Audit overview

Critical

No critical issues were found.

High

No high severity issues were found.

Medium

No medium severity issues were found.

Low

1. Missing Zero Address Validation

   In the tok_poa.sol contract, the transfer, and transferFrom functions arguments must be checked for 0x0. This also applies for the burnFrom and checkFrozenAddress methods.

   In 2_Owner.sol contract, the changeOwner() function should check its argument for missing zero address validation.

   Contracts:tok_poa.sol, 2_Owner.sol

   Function: transfer() between lines #338 - #372, transferFrom() between lines #374 - #411, burnFrom() between lines #449 - #454, checkFrozenAddress() between lines #208 - #241, changeOwner() between lines #39 - #47

   Recommendation: Implement a missing zero address check.

   Status: Fixed

2. Use of Hardcoded Values

   In the SafeMath library, INT256_MIN value uses hardcoded mathematics.

   In the POAToken contract totalSupply value uses hardcoded mathematics. getLiquidityFee, getEconomyFee, getStakeFee, getPrivateSaleFee, getAdditionalFee, getBurnFee methods use hardcoded values in mathematical operations too.

   Contracts:tok_poa.sol,

   Function:getLiquidityFee, getEconomyFee, getStakeFee, getPrivateSaleFee, getAdditionalFee, getBurnFee

   Recommendation: Move hard coded values to constants.

   Status: Reported

3. Floating Pragma

   The tok_poa.sol contract uses floating pragma ^0.7.0..

   Contracts:tok_poa.sol

   Function: -

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

   Status: Fixed

4. Missing Control for Existing Elements

   In excludeFromFee() and removeToPrivateSale() functions, instead of setting address to false, first check if the address is in these mappings first. If it is, remove it.

   Contracts:tok_poa.sol

   Function: excludeFromFee() between lines #243-247, removeToPrivateSale() between lines #262-267

   Recommendation: Implement above mentioned logic.

   Status: Fixed

5. Potential zero Parameter

   In transfer and transferFrom methods, the value parameter can be zero.

   Contracts:tok_poa.sol

   Function: transfer() between lines #338 - #372, transferFrom() between lines #374 - #411

   Recommendation: Implement checks.

   Status: Fixed

6. Boolean Equality

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

   Contracts:tok_poa.sol

   Function:POATokenContract.checkFrozenAddress(address,uint256) between lines #208-241, POATokenContract.transfer(address,uint256) between lines #338-372, POATokenContract.transferFrom(address,address,uint256) between lines #374-410, POATokenContract._burn(address,uint256,bool) between lines #429-439

   Recommendation: Avoid using boolean equalities.

   Status: Fixed

7. 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:token_poa.sol, 2_Owner.sol

   Function:burn(uint256) (token_poa.sol#445-447), burnFrom(address,uint256) (token_poa.sol#449-453), changeOwner(address) (2_Owner.sol#39-42), excludeFromFee(address[]) (token_poa.sol#243-247), includeInFee(address[]) (token_poa.sol#248-252), isExcludedFromFee(address) (token_poa.sol#253-255), addToPrivateSaleList(address[]) (token_poa.sol#257-261), removeToPrivateSaleList(address[]) (token_poa.sol#262-266)

   Recommendation: Aforementioned should be declared as *external*.

   Status: Fixed

8. Incorrect Solidity Version

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

   Contracts:token_poa.sol, 2_Owner.sol

   Function:-

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

   Status: Fixed

9. Misleading Function Name

   The checkFrozenAddress function name misleads people. It essentially checks if operations like transfer should be allowed or not.

   Contracts:token_poa.sol,

   Function: checkFrozenAddress between lines #208-242

   Recommendation: Rename this function.

   Status: Fixed

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

Recommendations

1. Separate vesting logic from the token contract. Make it a separate contract.

   Contracts: token_poa.sol

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.