HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Kyte.One
Date: March 15th 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; Vesting
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/Kyte-Research/kyte-core-contracts-Initial audit https://github.com/Kyte-Research/kyte-core-contracts/commit/b0806a6ac68c8b33992cc1ac2146074d23928d48 - Remediation check
Commit	d5685ad1298cafe91bed5d96f34496ec93c50b3c - Initial audit b0806a6ac68c8b33992cc1ac2146074d23928d48 - Remediation check
Technical Documentation	YES - https://www.airlyft.one
JS tests	YES
Website	https://www.airlyft.one
Timeline	01 FEB 2022 - 15 MARCH 2022
Changelog	02 MAY 2022 - Initial Audit 15 MARCH 2022 - Remediation 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 Kyte.One (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/Kyte-Research/kyte-core-contracts - Initial Audit
https://github.com/Kyte-Research/kyte-core-contracts/commit/b0806a6ac68c8b33992cc1ac2146074d23928d48 - Remediation Check

Commit:
d5685ad1298cafe91bed5d96f34496ec93c50b3c - Initial Audit
b0806a6ac68c8b33992cc1ac2146074d23928d48 - Remediation check

Technical Documentation: Yes - https://www.airlyft.one

JS tests: No

Contracts:
KTEToken.sol
TokenVesting.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

Score measurements details can be found in the corresponding section of the methodology.

Documentation quality

The customer provided superficial functional requirements and no technical requirements. The provided documents include tokenomics.Total Documentation Quality score is **3** out of **10**.

Code quality

Total CodeQuality score is **6** out of **10**. Redundant functions exist. Unit tests were provided.

Architecture quality

Architecture quality score is **6** out of **10**. Also one function did not care about the Gas optimization which led to high Gas costs.

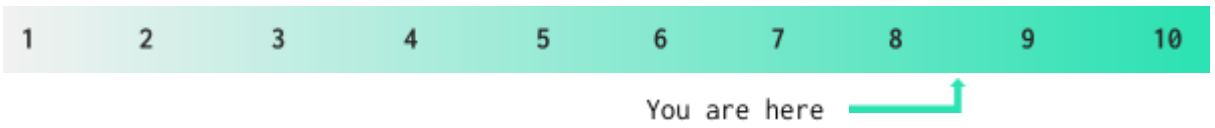
Security score

As a result of the audit, security engineers found **1** critical, **1** high, **1** medium, **7** low severity issues. Security score is **0** out of **10**. All found issues are displayed in the “Issues overview” section of the report.

As a result of the remediations review, Customers’ smart contracts contain **no** issues. Thus their security score is **10** out of **10**.

Summary

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



Notice: Owners of the contact can revoke vesting and withdraw all the tokens.

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

1. Token amount is not being updated after withdraw

   In the TokenVesting.sol contract, withdraw function allows the owner to get tokens from the contract. However, the contract's token balance is not being updated after this function's execution. This could lead to possible double spending. If owners want to revoke a vesting they should call corresponding revoke function instead of withdraw(), it is dangerous.

   Contracts:TokenVesting.sol

   Function: withdraw() between lines #173-179,

   Recommendation: Review and fix this logic.

   Status: Fixed

High

1. User may get no funds

   In the TokenVesting.sol contract, vesting does not guarantee a user will receive any token. There's no sign that project owner transfers any funds to this contract prior to vestings.

   Contracts:TokenVesting.sol

   Function: createVestingSchedule() between lines #76 - #119,

   Recommendation: Owner should transfer tokend to this contract in the vesting schedule creation.

   Status: Fixed

Medium

1. Potential High Gas Cost Risk

   In the TokenVesting.sol contract, getAllVestingScheduleForHolder function should be declared as external and if the underlying array size is too big, this function will fail due to high Gas consumption.

   Contracts:TokenVesting.sol

   Function: getAllVestingScheduleForHolder() between lines #290-306,

   Recommendation: Review and fix this logic.

   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 TokenVesting.sol contract, the createVestingSchedule function's _beneficiary parameter can be 0x0.

   In the TokenVesting.sol contract, the getAllVestingScheduleForHolder function's holder parameter can be 0x0.

   Contracts:TokenVesting.sol

   Function: createVestingSchedule() between lines #76 - #119, getAllVestingScheduleForHolder() between lines #290 - #310,

   Recommendation: Implement missing zero address check.

   Status: Fixed

2. Floating Pragma

   The TokenVesting.sol and KTEToken contracts use floating pragma ^0.8.0.

   Contracts: TokenVesting.sol, KTEToken.sol

   Function: -

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

   Status: Fixed

3. Missing Control for Existing Elements

   In the getAllVestingScheduleForHolder function of the TokenVesting.sol, there are no checks to confirm that the holder is a vester address.

   Contracts:TokenVesting.sol

   Function: getAllVestingScheduleForHolder() between lines #290-306

   Recommendation: Implement above mentioned logic.

   Status: Fixed

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

4. Potential zero Parameter

   In the withdraw and release functions of the TokenVesting.sol contract , *the amount* parameter can be zero.

   Contracts:TokenVesting.sol

   Function: withdraw() between lines #173-179, release() between lines #202-226

   Recommendation: Implement checks.

   Status: Fixed

5. Boolean Equality

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

   Contracts: TokenVesting.sol

   Function:revoke() between lines #140-162, _computeReleasableAmount() between lines #322-350

   Recommendation: Avoid using boolean equalities.

   Status: Fixed

6. 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:createVestingSchedule between lines #76-134, revoke between lines #140-162, withdraw between lines #168-174, releaseAllVested between lines #180-190, getVestingSchedulesCount between lines #227-229, computeReleasableAmount between lines #235-245, getWithdrawableAmount between lines 263-265, computeNextVestingScheduleIdForHolder between lines #270-280, getAllVestingScheduleForHolder between lines 285-301

   Recommendation: Aforementioned should be declared as *external*.

   Status: Fixed

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

7. Unused Modifier

   In the TokenVesting.sol contract, the modifier onlyIfVestingScheduleExists is defined but never used.

   Contracts: TokenVesting.sol,

   Function: onlyIfVestingScheduleExists between lines #29-35

   Recommendation: Remove this modifier.

   Status: Fixed

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.