HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Kasta
Date: March 07th, 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 Kasta.
Approved by	Andrew Matiukhin | CTO Hacken OU Evgeniy Bezuglyi | SC Audits Department Lead
Type	ERC-20 token vesting
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/kasta-io/token-vesting
Commit	7c5787fe2408624d95831652dd50cb2b540212f4
Technical Documentation	YES
JS tests	YES
Website	https://www.kasta.io/
Timeline	February 16, 2022 - March 7, 2022
Changelog	February 28, 2022 - Initial audit March 7, 2022 - Second Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Kasta (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 February 16th, 2022 - February 28th, 2022.

The second review was conducted on March 7th, 2022.

Scope

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

Repository:
https://github.com/kasta-io/token-vesting
Commit:
7c5787fe2408624d95831652dd50cb2b540212f4
Technical Documentation: Yes
JS tests: Yes
Contracts:
KastaVesting.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 project has good documentation with functional and technical requirements. The score is **10** out of **10**. Weight in total score is **1**.

Code quality

The code follows official language style guides and is covered with unit tests. Most of the code follows those guides, the score is **10** out of **10**. Weight in total score is **1**.

Architecture quality

Smart contract of the project follows the best practices.

Clean and clear architecture, the score is **10** out of **10**. Weight in total score is **1**.

Security score

As a result of the audit, security engineers found **1** medium and **1** low severity issue. Security score is **7.5** out of **10**. All found issues are displayed in the “Issues overview” section of the report. Weight in total score is **7**.

After the second audit the code has **1** medium severity issue.

Summary

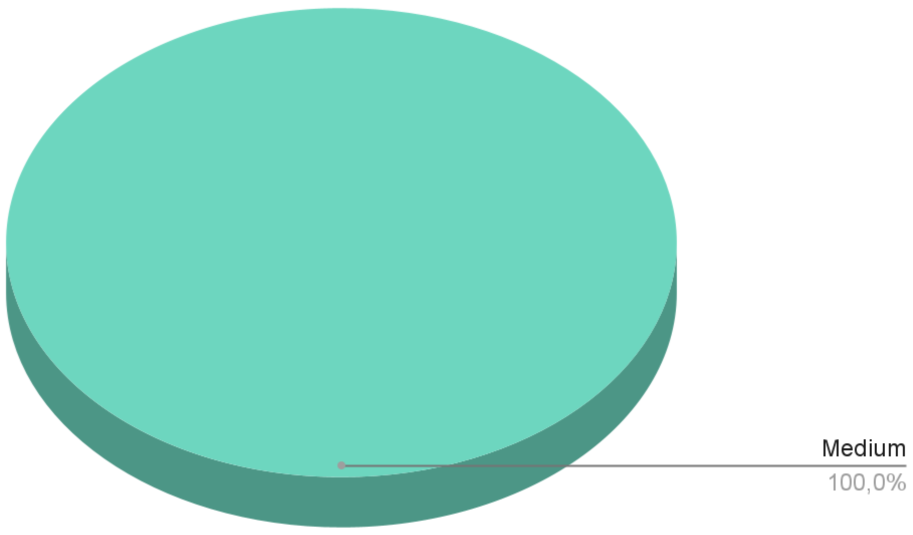
According to the assessment, the Customer's smart has the following score: **8.25**

1
2
3
4
5
6
7
8
9
10
You are here

Notices

1. The KastaVesting contract has a `revokeSchedule` function which sends available unclaimed tokens to the admin.
2. All claims could be paused by owners.
3. The KastaVesting contract allows creating vesting with the start date in the past.

Graph 1. The distribution of vulnerabilities after the audit.



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

Audit overview

🟥🟥🟥🟥 Critical

No critical severity issues were found.

🟨🟨🟨 High

No high severity issues were found.

🟩🟩 Medium

The code contains redundant start date limitations in days. As a result, the contract would be inoperable after January 1st, 2030.

Contracts: KastaVesting.sol

Recommendation: validate the start date without limitations in the future.

Status: Acknowledged

🟦 Low

\`vested\` amount calculation performs division before multiplication which potentially may cause rounding issues.

Contracts: KastaVesting.sol

Function: _getAvailableAmount, _getNotVestedAmount

Recommendation: perform multiplication before division.

Status: Fixed

Recommendations

1. Simplify the vesting logic and make it more flexible by removing calculations in days and using seconds instead. Such an approach is more flexible and lowers the code complexity.

   Contracts: KastaVesting.sol

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.