HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Metacloud
Date: March 29th, 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 Metacloud.
Approved By	Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type of Contracts	ERC20 token; Crowdsale; Vesting
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	http://meta-cloud.io
Timeline	18.03.2022 - 29.03.2022
Changelog	23.03.2022 - Initial Review 29.03.2022 - Revision

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

Findings 8

Disclaimers 11

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

Introduction

Hacken OÜ (Consultant) was contracted by Metacloud (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/Safegramhub/metacloud-ico-contract
Commit:
0fb66762def4ce0635139cf13436d09168f35e0b
Technical Documentation: Yes
JS tests: Yes
Contracts:
./contracts/token/Context.sol (Removed in 0fb6676)
./contracts/token/ERC20.sol (Removed in 0fb6676)
./contracts/token/IERC20.sol (Removed in 0fb6676)
./contracts/token/MetaCloud.sol
./contracts/token/Ownable.sol (Removed in 0fb6676)
./contracts/token/SafeMath.sol (Removed in 0fb6676)
./contracts/MetaCloudCrowdsale.sol
./contracts/TokensVesting.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 ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ EIP standards violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository 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 ▪ Kill-Switch Mechanism

Executive Summary

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

Documentation quality

The Customer provided general functional requirements and no technical requirements. Total Documentation Quality score is **6** out of **10**.

Code quality

The total CodeQuality score is **7** out of **10**. Some unit tests were provided, but code coverage is about 40%.

Architecture quality

The architecture quality score is **10** out of **10**. The project has clean and clear architecture.

Security score

As a result of the audit, security engineers found **2** critical, **4** high, **1** medium, and **1** low severity issues.

As a result of the revision, security engineers found **1** **new** high severity issue. All previously found issues were fixed.

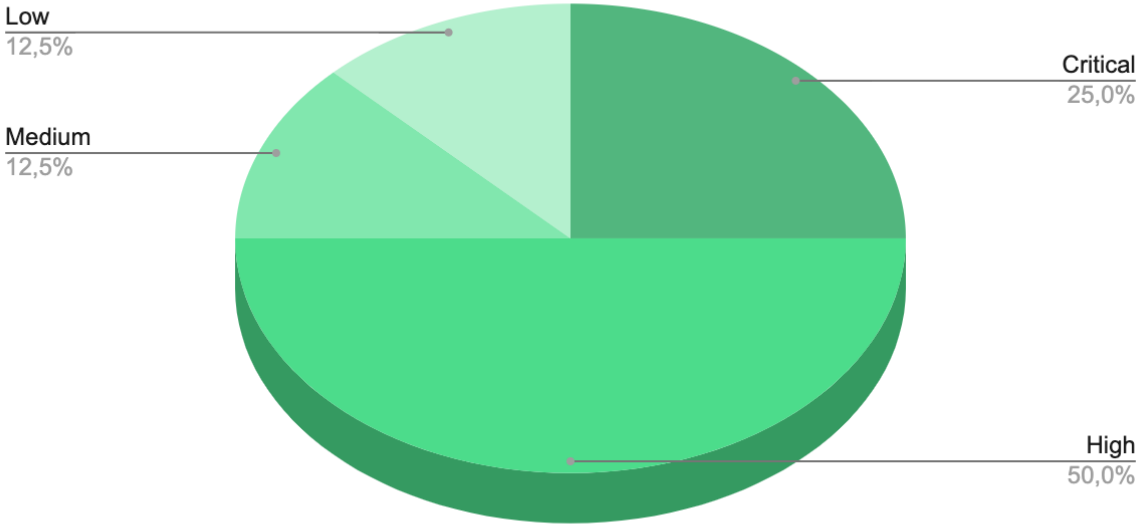
The security score is **5** out of **10**. All found issues are displayed in the “Issues overview” section.

Summary

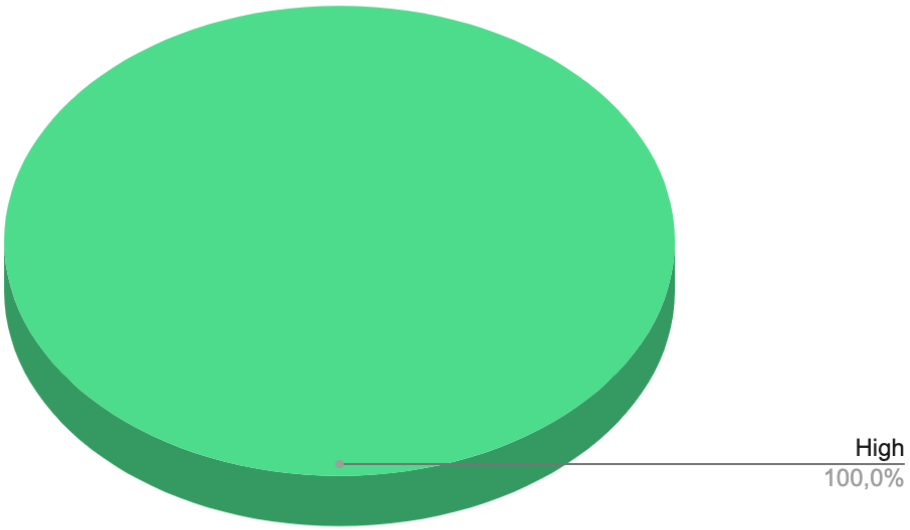
According to the assessment, the Customer's smart contract has the following score: **5.8**

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

Graph 1. The distribution of vulnerabilities after the audit.



Graph 2. The distribution of vulnerabilities after the revision.



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

Findings

Critical

1. **Error in calculations.**

   releasesCount is 10 and is calculated by dividing by 100, resulting in tokensPerRelease always being 0. This function returns a non-zero value only when the vesting is completely over.

   This leads to funds stuck on the contract until the end of vesting.

   Contracts: TokensVesting.sol

   Function: calculateVestedTokens

   Recommendation: fix calculations.

   Status: Fixed (Revised commit: 0fb6676)

2. **Wrong logic for selling for BNB.**

   All of these functions accept BNB as payment for the Cloud, but the rate is the same as for BUSD, and the buyer still has to pay BUSD even if BNB is paid.

   This functionality is broken.

   Contracts: MetaCloudCrowdsale.sol

   Functions: buy, fallback, receive

   Recommendation: if the ability to pay in BNB, change the code to make it the same price as BUSD, otherwise remove it.

   Status: Fixed (Revised commit: 0fb6676)

High

1. **Highly permissive owner access.**

   All token transfers can be stopped by owners at any time by using closeCrowdsale(false). Only the crowdsale address can run this function, but owners can change the crowdsale address.

   Contract: MetaCloud.sol

   Recommendation: restrict the ability to stop transfers or remove this ability.

   Status: Fixed (Revised commit: 0fb6676)

2. **Wrong BUSD address.**

   The BUSD token address hardcoded into the contract is an Ethereum BUSD address, but according to the documentation, this contract must operate on the BSC network.

   Contract: MetaCloudCrowdsale.sol

   Recommendation: use the correct BUSD address for the BSC network.

Status: Fixed (Revised commit: 0fb6676)

3. **Highly permissive owner access.**

   The startTheVesting function can be called multiple times. Whenever it was called, the vesting period changed. Only the crowdsale address can run this function, but owners can change the crowdsale address.

   The ability to change the vesting period may result in buyers not receiving their tokens or receiving them much later than the stated deadline.

   Contract: TokensVesting.sol

   Function: startTheVesting

   Recommendation: remove the ability to call the startTheVesting function more than once.

   Status: Fixed (Revised commit: 0fb6676)

4. **Costly loop.**

   The startTheVesting function uses a loop to set the start and end of the vesting period for each buyer. The number of loop iterations depends on the number of buyers. If the number of buyers is large (a specific value can be calculated or determined empirically), then the transaction will go beyond the gas limit and will be reverted. Even if the transaction does not go beyond the gas limit, it will still be costly.

   In the worst case, all buyer's funds will be stuck on the contract since vesting will be impossible to start.

   Contracts: TokensVesting.sol

   Function: startTheVesting

   Recommendation: according to the logic of the contract, all buyers have the same vesting period, so can be stored these dates globally for the entire contract and not for each user separately. Remove the loop.

   Status: Fixed (Revised commit: 0fb6676)

5. **Logic error.**

   At the end of the first phase of sales, the number of unsold tokens may be less than possible to buy according to the minimumBuyAmount.

   In this case, no one will be able to buy the remaining tokens of the first phase, and the second phase will never begin.

   Contract: MetaCloudCrowdsale.sol

   Function: buy

   Recommendation: fix this edge case.

   Status: New

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

Medium

1. Unsafe token transfers.

Token transfer functions do not throw an error on failed transfer.

This can lead to situations where the state of the contract has changed, but no tokens have been transferred.

Contracts: TokensVesting.sol, MetaCloudCrowdsale.sol, MetaCloud.sol

Recommendation: add some wrappers around ERC20 operations that throw on failure(e.g. SafeERC20).

Status: Fixed (Revised commit: 0fb6676)

Low

1. Unused function.

The _setupDecimals function is never used.

Contracts: ERC20.sol

Function: _setupDecimals

Recommendation: remove unused function.

Status: Fixed (Revised commit: 0fb6676)

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.