SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Ground Zero
Date: May 03rd, 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 Ground Zero.
Approved By	Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type of Contracts	IDO
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://ground-zero.io
Timeline	23.03.2022 - 03.05.2022
Changelog	29.03.2022 - Initial Review 19.04.2022 - Remediation 03.05.2022 - Second Remediation

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Ground Zero (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:

Initial review

Deployed contracts:
https://testnet.bscscan.com/address/0x1936367815e25Bf15F956AF37CcB28C21e65383b
Technical Documentation: Yes
(https://drive.google.com/file/d/1BTMeaesdBa4LDlPTJ-sXAsvgy_ylrrle/view)
JS tests: No
Contracts:
IDO.sol

Second review

Deployed contracts:
https://testnet.bscscan.com/address/0x76c7C9E9c478E52e4303843129144Cc463721096
Technical Documentation: Yes
(https://drive.google.com/file/d/1BTMeaesdBa4LDlPTJ-sXAsvgy_ylrrle/view)
JS tests: No
Contracts:
IDO.sol

Third review

Repository:
https://github.com/GotBit/GroundZero-IDO/tree/contracts-stable
Commit:
122c8b550ab77d356bcc3faeb47bb7a6c4f1f1f2
Technical Documentation: Yes
JS tests: No
Contracts:
contracts/contracts/IDO.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
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 technical requirements and no functional requirements. The total Documentation Quality score is **5** out of **10**.

Code quality

The total CodeQuality score is **9** out of **10**. Unit tests were provided, but some tests failed.

Architecture quality

The architecture of the contract is clear. The architecture quality score is **10** out of **10**.

Security score

As a result of the audit, security engineers found **1** critical, **4** high, and **3** low severity issues. The security score is **0** out of **10**. All found issues are displayed in the “Issues overview” section.

As a result of the second review, security engineers found **1** new high, **1** medium, and **1** low severity issue. The code contains **1** high, **1** medium, and **2** low severity issues. The security score is **5** out of **10**.

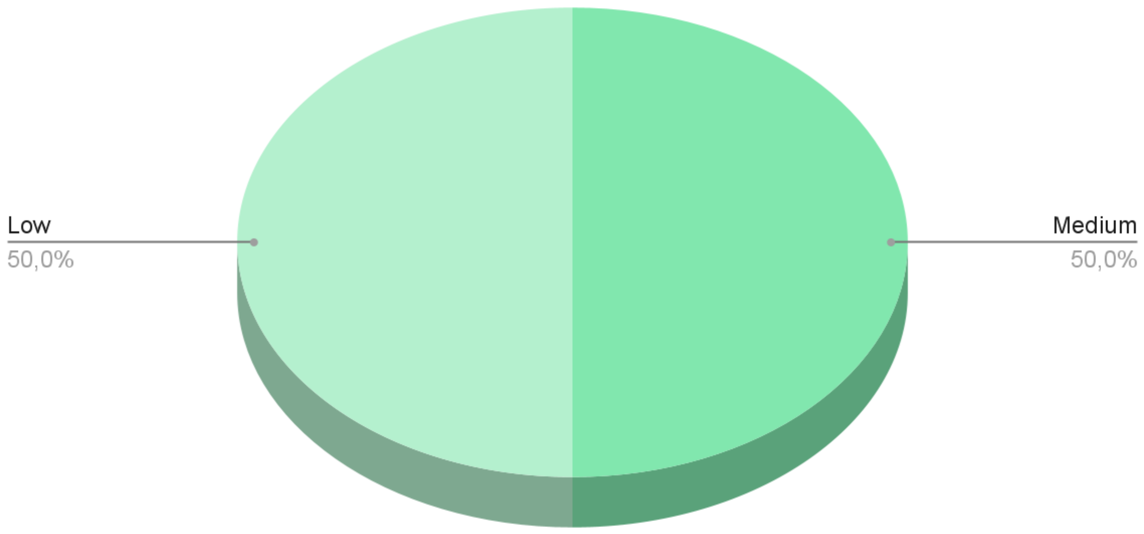
As a result of the third review, security engineers found **1** new medium severity issue. The code contains **1** medium and **1** low severity issues. The security score is **10** out of **10**.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.4**

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

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 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

The claimedAmount does not change on token claiming.

When claiming tokens, the claimedAmount variable always remains 0.

This means that the user can withdraw tokens from the contract without restrictions on the number.

Contracts:IDO.sol

Function: claim

Recommendation: add to the claimedAmount variable the amount of tokens withdrawn each time.

Status: Fixed (Second review)

🟨🟨🟨 High

1. **Owner`s ability to withdraw tokens from the contract.**

   The owner can withdraw tokens that belong to users.

   Contracts: IDO.sol

   Function: claimTheInvestments

   Recommendation: take away from the owner the right to withdraw any tokens.

   Status: Fixed (Second review)

2. **Owner`s ability to change IDO data anytime.**

   The owner can call the setIDO function at any time.

   This can block the withdrawal of tokens by changing the finished variable or changing other IDO data without the consent of users.

   Contracts: IDO.sol

   Function: setIDO

   Recommendation: take away from the owner the right to change IDO data anytime.

   Status: Fixed (Second review)

3. **Token transfer results are not validated.**

   The results of all token transfers in the contract are not processed.

   This can lead to erroneous changes in state variables.

   Contracts: IDO.sol

   Function: claim, claimTheInvestments

Recommendation: process the result of token transfers.

Status: Fixed (Second review)

4. **Change of state variables after withdrawal of tokens.**

   The variables claimedLastTime or hasClaimed change only after the token transfer.

   This can lead to a re-entrancy attack, and the user will be able to withdraw tokens regardless of time.

   Contracts:IDO.sol

   Function: claim

   Recommendation: Change the claimedLastTime or hasClaimed before transferring tokens or add “nonReentrant” modifier.

   Status: Fixed (Second review)

5. **Highly permissive owner access**

   The functionality allows the owner to change the allowed token rate anytime.

   This can result in users receiving fewer GZT tokens than they expected.

   Contracts: IDO.sol

   Function: allowToken, participate

   Recommendation: in the “participate” function validate the amount of GZT tokens that users expect to match the “convertedAmount” value.

   Status: Fixed (Third review)

■■ Medium

1. **Checks-Effects-Interactions pattern violation.**

   The state variables are changed after the tokens transfer.

   Performing transfer after changing the state protects the contract from re-entrance and race conditions if ambiguous token implementation is used.

   Contracts: IDO.sol

   Function: claim, participate

   Recommendation: perform transfer after the state is changed.

   Status: Fixed (Third review)

2. **Tests failing.**

   8 tests fail with the reason “participate: ido is not started yet”.

   All the tests should be passed.

Contracts: IDO.sol

Function: -

Recommendation: ensure that all the tests are passed.

Status: New

Low

1. Unlocked pragma

Contracts with unlocked pragmas may be deployed by the latest compiler, which may have higher risks of undiscovered bugs.

Contracts: IDO.sol

Recommendation: lock pragmas to a specific compiler version.

Status: Acknowledged

2. No event on token claiming.

It is recommended to emit events if the contract’s state is changed.

Contracts: IDO.sol

Functions: claim

Recommendation: emit an event on token claiming.

Status: Fixed (Second review)

3. Unused imports

Imported
@openzeppelin/contracts/token/ERC1155/presets/ERC1155PresetMinterPauser.sol,
@openzeppelin/contracts/token/ERC20/presets/ERC20PresetMinterPauser.sol,
hardhat/console.sol are not used.

Contracts: IDO.sol

Recommendation: remove unused imports.

Status: Fixed (Second review)

4. Never emitted event

The contract has the “WithdrawedGZT” event that is never emitted.

Contracts: IDO.sol

Recommendation: emit “WithdrawedGZT” event on the GZT withdrawing.

Status: Fixed (Third review)

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.