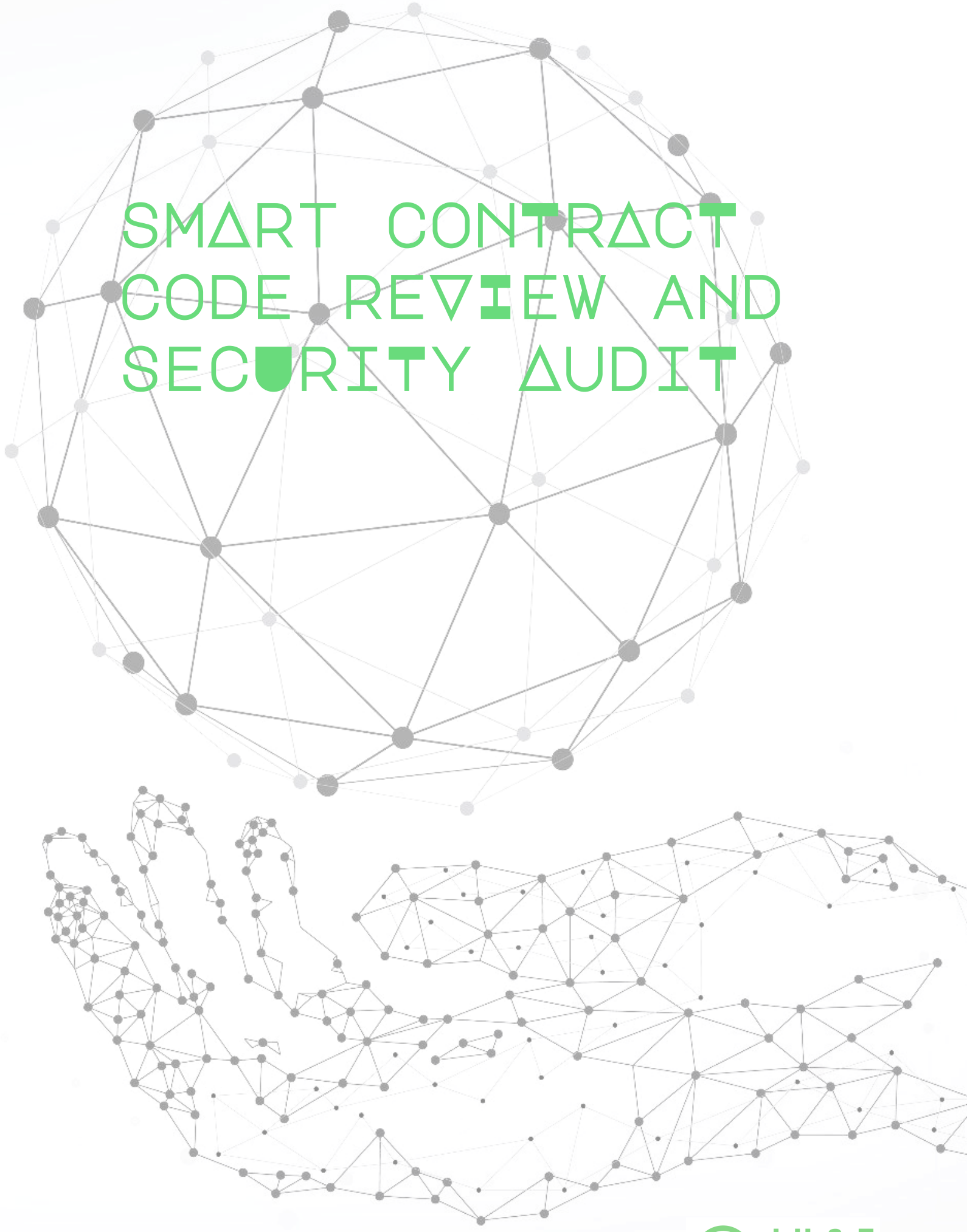
SMART CONTRACT CODE REVIEW AND SECURITY AUDIT



Customer: BlockAgents
Date: Feb 25th, 2022

JUST AUDIT

This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed — upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for BlockAgents
Type	ERC721 token; Merkle tree whitelist
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	github.com/Cojodi/blockagents_contracts/
Commit	d5e480afd30c6996297130e8f4163e33
Technical Documentation	No
Automated Tests	Yes, in code
Website	blockagents.io
Timeline	Feb 23rd, 2022 - Feb 25th, 2022
	Feb 23rd, 2022 - Initial Audit Feb 24th, 2022 - Second Review

Table of Contents

Introduction

JustAudit.me was contracted by BlockAgents.io (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 23rd, 2022 and Feb 25th, 2022.

The second review was provided on February 24th , 2022.

Scope

For the audit, we were provided the following:

Github archive: github.com/Cojodi/blockagents_contracts/
Md5 hash: d5e480afd30c6996297130e8f4163e33
Technical Documentation: No
JS tests: No

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	Items
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

Summary

According to the assessment, the Customer's smart contracts are **well-secured.**

Insecure	Poorly Secured	Secure	Well-secured

You are here ↑

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither.

All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found 1 medium and 1 low severity issues.

After the second review security engineers found that most of the issues were fixed and there were no security issues left in the code.

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

1. Multiple calls executed in same transactions

   This call is executed following another call within the same transaction. It is possible that the call never gets executed if a prior call fails permanently. This might be caused intentionally by a malicious callee.

   Contracts: BlockAgentsMintpass.sol

   Recommendation: If possible, refactor the code such that each transaction only executes one external call or make sure that all callees can be trusted (i.e. they're part of your own codebase).

   Status: Acknowledged. The function is called by the project owners.

❐ Low

1. A floating pragma is set.

   It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

   Contracts: Whole codebase

   Recommendation: Set a fixed pragma

   Status: Acknowledged. Set fixed version.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found **no security issues.**

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, the Source Code compilation, deployment and functionality (performing the intended functions).

Because the total number of test cases is unlimited, 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, bugfree 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 security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.