

HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Doge Universe
Date: September 7th, 2021



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 Doge Universe.
Approved by	Andrew Matiukhin CTO Hacken OU
Type	ERC20 token; Transfer controller
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Contract	https://rinkeby.etherscan.io/address/0x9f532E3A08857118Bc2EbA769d261743850E479E#code
Technical Documentation	NO
JS tests	NO
Timeline	1 SEPTEMBER 2021 - 7 SEPTEMBER 2021
Changelog	7 SEPTEMBER 2021 - INITIAL AUDIT



Table of contents

Introduction	4
Scope	4
Executive Summary	5
Severity Definitions	7
Audit overview	8
Conclusion	8
Disclaimers	10



Introduction

Hacken OÜ (Consultant) was contracted by Doge Universe (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contract and its code review conducted between September 1st, 2021 - September 7th, 2021.

Scope

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

Contract:

<https://rinkeby.etherscan.io/address/0x9f532E3A08857118Bc2EbA769d261743850E479>

Technical Documentation: No

JS tests: No

File:

[SPACExDOGE.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	<ul style="list-style-type: none">ReentrancyOwnership TakeoverTimestamp DependenceGas Limit and LoopsDoS with (Unexpected) ThrowDoS with Block Gas LimitTransaction-Ordering DependenceStyle guide violationCostly LoopERC20 API violationUnchecked external callUnchecked mathUnsafe type inferenceImplicit visibility levelDeployment ConsistencyRepository ConsistencyData Consistency

Functional review	<ul style="list-style-type: none"> ▪ 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

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



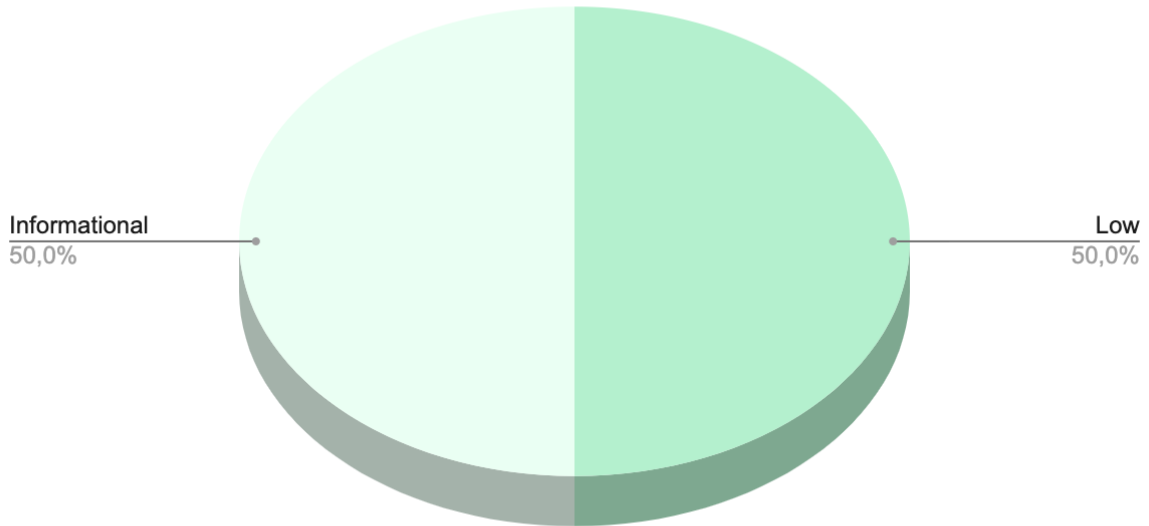
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **1** low severity and **1** informational issue.

Notice:

- There are no functional tests. We can't be responsible for issues with Business logic.
- There is no code documentation.

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 issues were found.

■ ■ ■ High

No high severity issues were found.

■ ■ Medium

No medium severity issues were found.

■ Low

There are several duplicates checking the transfer address not to be equals with the router address. Move checking transfer from uniswapV2Pair to require.

■ Information

`_getCurrentSupply()` function result can be cached to field value to remove double computation and decrease gas usage.



Conclusion

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

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

As a result of the audit, security engineers found **1** low severity and **1** informational issue.

Notice:

- There are no functional tests. We can't be responsible for issues with Business logic.
- There is no code documentation.



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, 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 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.