HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Madhouse
Date: August 20ᵗʰ, 2021

Document

Name	Smart Contract Code Review and Security Analysis Report for Madhouse.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	BEP20 token with Fees
Platform	Binance Smart Chain / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Zip archive	bep20-madhouse-token-main.zip
Technical Documentation	NO
JS tests	NO
Timeline	06 AUGUST 2021 – 20 AUGUST 2021
Changelog	12 AUGUST 2021 – INITIAL AUDIT 20 AUGUST 2021 – SECOND REVIEW

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Madhouse (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 August 6th, 2021 - August 12th, 2021. The second review conducted on August 20th, 2021.

Scope

The scope of the project is smart contracts in the repository:
Zip archive:
      bep20-madhouse-token-main.zip
MD5 hash:
      1a093443f6efde48dfdf949904058a3a
Technical Documentation: No
JS tests: No
Contracts:
      MadhouseToken.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

According to the assessment, the Customer's smart contract is well-secured.

Insecure	Poor secured	Secured	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 **6** low severity issues.

After the second review security engineers found no issues.

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

1. No events on fees changes

   Changing fees should emit events so that could be easily tracked off-chain.

   Recommendation: Please consider emitting events on changing fees.

   Fixed before the second review.

2. Unused state variable

   Contract <u>MadhouseToken</u> has a state variable **uniswapV2Router** which is never read in the code. Only writing is done in <u>constructor</u> and <u>changeRouterVersion</u> but it's never accessed for the reading which means it is just burning gas.

   Recommendation: Please consider removing the **uniswapV2Router** state variable.

   Fixed before the second review.

3. Unused private method

   Contract <u>MadhouseToken</u> has a private method **<u>transferToAddressETH</u>** which is never called in the code.

   Recommendation: Please consider removing this method.

   Fixed before the second review.

4. Contracts that lock Ether

   Contract <u>MadhouseToken</u> has a payable function but without a withdrawal capacity.

   Recommendation: Remove the payable attribute or add a withdraw function.

   Fixed before the second review.

5. State variables that could be declared constant

   Constant state variables should be declared constant to save gas.

Recommendation: Add the **constant** attributes to state variables that never change.

Fixed before the second review.

6. Public functions that could be declared external

   public functions that are never called by the contract should be declared **external** to save gas.

   Recommendation: Use the **external** attribute for functions never called from the contract.

   Fixed before the second review.

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 **6** low severity issues.

After the second review security engineers found no issues.

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.