HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Shibu Foundation
Date: December 01st, 2021

Document

Name	Smart Contract Code Review and Security Analysis Report for Shibu Foundation.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 tokens
Platform	Binance / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/Shibu-Foundation/shibu-contracts
Commit	f7a203b36fbe3f8c7c11f8d26a48011b803115b8
Technical Documentation	NO
JS tests	NO
Website	shibu.foundation
Timeline	22 NOVEMBER 2021 – 01 DECEMBER 2021
Changelog	26 NOVEMBER 2021 – INITIAL AUDIT 01 DECEMBER 2021 - SECOND REVIEW

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Shibu Foundation (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 November 22ⁿᵈ, 2021 - November 26ᵗʰ, 2021.

Second review conducted on December 1ˢᵗ, 2021.

Scope

The scope of the project is smart contracts in the repository:
Repository:
  https://github.com/Shibu-Foundation/shibu-contracts
Commit:
  f7a203b36fbe3f8c7c11f8d26a48011b803115b8
**Technical Documentation:** No
JS tests: No
Contracts:
  Shibu.sol
  ShibuDividendTracker.sol
  interfaces/DividendPayingTokenOptionalInterface.sol
  interfaces/ICoinswap.sol
  interfaces/DividendPayingTokenInterface.sol
  interfaces/IBEP20.sol
  libraries/IterableMapping.sol
  libraries/Ownable.sol
  libraries/Context.sol
  libraries/BEP20.sol
  libraries/DividendPayingToken.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 contracts are secured.

| Insecure	Poor secured	Secured	Well-secured

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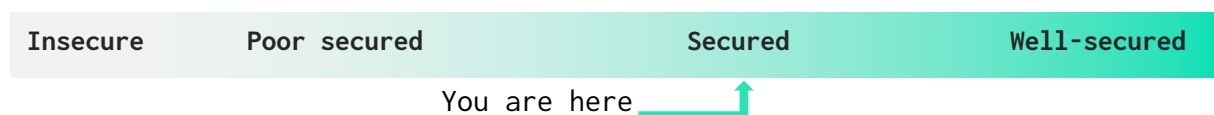
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview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

As a result of the audit, security engineers found **1** medium and **5** low severity issues.

After the second review, security engineers found **1** medium severity issue.

Graph 1. The distribution of vulnerabilities after the audit.

Medium
100,0%
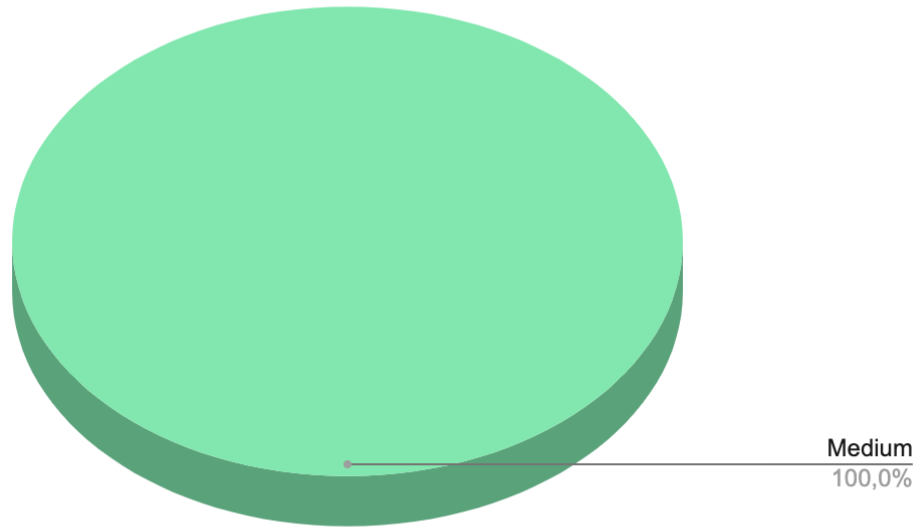
Severity Definitions

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 tests were provided.

It's recommended to cover all non-trivial contracts with tests.

The recommended coverage is minimum 95% for branches, while it should be definitely 100% for the main logic contracts.

■ Low

1. Boolean equality

   Boolean constants can be used directly and do not need to be compared to true or false.

   Contracts: ShibuDividendTracker.sol

   Functions: excludeFromDividends

   Recommendation: remove the equality to the boolean constant.

   Status: fixed

2. Missing event for changing *minimumTokenBalanceForDividends, BUSDRewardsFee, charityFee, liquidityFee, autoBoost, burnFee, swapTokensAtAmount, maxWalletBalance, _isBlacklisted[], maxTxAmount, swapEnabled, autoBoostEnabled, autoBoostThreshold*

   Contracts: ShibuDividendTracker.sol, Shibu.sol

   Functions: setMinimumBalanceForRewards, setFees, setSwapTokensAtAmount, setMaxWalletBalance, setBlacklistAccount, setMaxTxAmount, setSwapEnabled, setAutoBoostEnabled, setAutoBoostThreshold

   Changing critical values should be followed by the event emitting for better tracking off-chain.

   Recommendation: Please emit events on the critical values changing.

   Status: fixed

3. Using SafeMath in Solidity >= 0.8.0

Starting solidity version 0.8.0 arithmetic operations revert on underflow and overflow. There's no more need to assert the result of operations.

Contracts: Shibu.sol, DividendPayingToken.sol, ShibuDividendTracker.sol

Recommendation: Please avoid using assert for arithmetic operations.

Status: fixed

4. State variables that could be declared constant

Constant state variables should be declared constant to save gas.

Contracts: Shibu.sol, DividendPayingToken.sol

Variables: BUSD

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

Status: fixed

5. A public function that could be declared external.

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

Contracts: Shibu.sol, DividendPayingToken.sol, ShibuDividendTracker.sol

Functions: getAccountAtIndex, process, distributeBUSDDividends, dividendOf, withdrawnDividendOf, updateDividendTracker, updateRouter, excludeMultipleAccountsFromFees, setAutomatedMarketMakerPair, updateGasForProcessing, isExcludedFromFees, dividendTokenBalanceOf

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

Status: fixed

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 **1** medium and **5** low severity issues.

After the second review, security engineers found **1** medium severity issue.

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.