



SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Metashi

Date: January 10th, 2022



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 Metashi.
Approved by	Bert Lawrence CTO AuditSolidTech LLC
Type	ERC20 token
Platform	Binance / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	Github.com/MetashiOfficial/Contract
Technical Documentation	NO
JS tests	YES
Website	metashi.net
Timeline	07 JANUARY 2022 - 10 JANUARY 2022
Changelog	10 JANUARY 2022 - INITIAL AUDIT



Audit Solid Tech
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@auditsolid.tech

Table of contents

Introduction	4
Scope	4
Executive Summary	5
Severity Definitions	7
Audit overview	8
Conclusion	9
Disclaimers	10



Introduction

AuditSolidTech LLC (Consultant) was contracted by Metashi (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 January 7th, 2022 – January 10th, 2022.

Scope

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

Repository:

[Github.com/MetashiOfficial/Contract](https://github.com/MetashiOfficial/Contract)

Technical Documentation: No

JS tests: Yes, in the repository

Contracts:

[METASHI.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">▪ 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



	<div>Repository Consistency</div> <div>Data Consistency</div>
Functional review	<div><div>▪ Business Logics Review</div><div>▪ Functionality Checks</div><div>▪ Access Control & Authorization</div><div>▪ Escrow manipulation</div><div>▪ Token Supply manipulation</div><div>▪ Assets integrity</div><div>▪ User Balances manipulation</div><div>▪ Data Consistency manipulation</div><div>▪ Kill-Switch Mechanism</div><div>▪ Operation Trails & Event Generation</div></div>

Executive Summary

According to the assessment, the Customer’s smart contracts are 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 3 low severity issue.



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. Missing event for changing liquidityFee , burnFee , teamBuyFee
teamSellFee , lotteryFee

Contracts: METASHI.sol

Functions: setLiquidityFee, setBurnFee, setTeamBuyFee,
setTeamSellFee, setLotteryFee

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

Recommendation: Please emit events on the critical values
changing.

2. State variables that could be declared constant

Constant state variables should be declared constant to save gas.

Contracts: METASHI.sol

Variables: deadAddress, tSupply

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

3. 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: METASHI.sol

Functions: withdraw, info, setAutomatedMarketMakerPair

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



Audit Solid Tech
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@auditsolid.tech

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 3 low severity issue.



Audit Solid Tech
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@auditsolid.tech

Disclaimers

AuditSolid 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.