**MuslimCoins Smart Contract Initial Audit Report**

[**Overview**](#_6l10zwfie74w) **2**

[Scope of Audit](#_la4tz2m9f16g) 2

[Check Vulnerabilities](#_fk11hakqdckb) 2

[**Techniques and Methods**](#_kmnjskexwpfd) **2**

[Issue Categories](#_2p6a8gpqk15r) 3

[Number of security issues per severity.](#_1ksv4uv) 4

[**Introduction**](#_41paf4h67ss0) **5**

[**Issues Found – Code Review / Manual Testing**](#_jnyy7e310pds) **5**

[High Severity Issues](#_fpa7vs16wijg) 5

[Medium Severity Issues](#_ihaplnhuvava) 5

[Low Severity Issues](#_3xubtpht259r) 5

[[L1] Missing zero address validation](#_jqeqb6wj24j9) 5

[Informational Issues](#_urpasvdhsbj6) 6

[[INF1] Missing comments and description:](#_ejbskerf8ukp) 6

[[INF2] Same execution for two different cases can be merged](#_3zcu8hgfhm9) 6

[[INF3] Less meaningful variable:](#_3x17wd7se9vj) 7

[[INF4] Variable defined but never used](#_a4z4ct502mz9) 7

[[INF5] Public methods only being used externally](#_jmhyvgcj81r3) 7

[[INF6] Constant calculations in the contract](#_7kz1vsz8vvf2) 8

[[INF7] Use a constructor to set addresses](#_eg2zw31i6nbh) 8

[[INF8] Getters should be at the bottom of the contract](#_tahhiir2tzo3) 9

[[INF9] Length calculation within the loop](#_tqms3u5xzm00) 9

[[INF10] Inconsistent error messages](#_uby07kylicr7) 9

[[INF11] Naming Conventions](#_wkx5y5fgwmq) 10

[[INF12] Presence of not implemented payable method](#_wkhqjdry56we) 10

[**Binance Testnet Test Contract**](#_t1tl3cno0wb3) **10**

[**Functional Tests**](#_q7wlj1ombaqi) **10**

[**Automated Tests**](#_1hif0py24gev) **11**

[Results:](#_nnxarl1y1et8) 12

[**Closing Summary**](#_wt0t7ihkiqxh) **13**

[**Disclaimer**](#_i7nr8bi7b69r) **13**

## **Overview**

## **Scope of Audit**

The scope of this audit was to analyze and document the MuslimCoins Token smart contract codebase for quality, security, and correctness.

## **Check Vulnerabilities**

* Re-entrancy
* Timestamp Dependence
* Gas Limit and Loops
* DoS with Block Gas Limit
* Transaction-Ordering Dependence
* Use of tx.origin
* Exception disorder
* Gasless send
* Balance equality
* Byte array
* Transfer forwards all gas
* ERC20 API violation
* Malicious libraries
* Compiler version not fixed
* Redundant fallback function
* Send instead of transfer
* Style guide violation
* Unchecked external call
* Unchecked math
* Unsafe type inference
* Implicit visibility level

## **Techniques and Methods**

Throughout the audit of smart contracts, care was taken to ensure:

* The overall quality of code.
* Use of best practices.
* Code documentation and comments match logic and expected behavior.
* Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
* Implementation of ERC-20 token standards.
* Efficient use of gas.
* Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

**Structural Analysis**

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

**Static Analysis**

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

**Code Review / Manual Analysis**

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

**Gas Consumption**

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

**Tools and Platforms used for Audit**

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

### **Issue Categories**

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

**High Severity Issues**

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract’s performance or functionality, and we recommend these issues be fixed before moving to a live environment.

**Medium Severity Issues**

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

**Low Severity Issues**

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

**Informational** **Issues**

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

## **Number of security issues per severity.**

## 

| **TYPE** | **HIGH** | **MEDIUM** | **LOW** | **INFORMATIONAL** |
| --- | --- | --- | --- | --- |
| **Open** | **0** | **0** | **1** | **12** |
| **Acknowledged** | **0** | **0** | **0** | **0** |
| **Closed** | **0** | **0** | **0** | **0** |

# **Introduction**

During the period of **November 24, 2021 to November 27, 2021** - QuillAudits Team performed a security audit for **MuslimCoins** smart contracts.

The code for the audit was taken from following the official link**:**

**Codebase:** [0x8B93585978B81E4FC0aE063fe526dBfBE9B8D42D](https://bscscan.com/address/0x8B93585978B81E4FC0aE063fe526dBfBE9B8D42D#code)

| **Version Number** | **Date** | **Contract address** | **Network** |
| --- | --- | --- | --- |
| 01 | 24 November | [0x8B93585978B81E4FC0aE063fe526dBfBE9B8D42D](https://bscscan.com/address/0x8B93585978B81E4FC0aE063fe526dBfBE9B8D42D#code) | Binance |

**Contract - MuslimCoins**

# **Issues Found – Code Review / Manual Testing**

## **High Severity Issues**

**None**

## **Medium Severity Issues**

**None**

## **Low Severity Issues**

### **[L1] Missing zero address validation**

Missing zero address check for to address in the following methods:

* *\_transferBothExcluded()*
* *\_approve()*
* *\_transfer()*
* *\_tokenTransfer()*

**Recommendation:** Add a ‘require’ to check **to** address != address(0)

**Status**: **Open**

## **Informational Issues**

### **[INF1] Missing comments and description:**

Comments and Description of the methods and the variables are missing, it's hard to read and understand the purpose of the variables and the methods in context of the whole picture

**Recommendation**: Consider adding NatSpec format comments for the  
 comments and state variables

**Status**: **Open**

### **[INF2] Same execution for two different cases can be merged**

if (\_isExcluded[sender] && !\_isExcluded[recipient]) {

\_transferFromExcluded(sender, recipient, amount);

} else if (!\_isExcluded[sender] && \_isExcluded[recipient]) {

\_transferToExcluded(sender, recipient, amount);

} else if (!\_isExcluded[sender] && !\_isExcluded[recipient]) {

\_transferStandard(sender, recipient, amount);

} else if (\_isExcluded[sender] && \_isExcluded[recipient]) {

\_transferBothExcluded(sender, recipient, amount);

} else {

\_transferStandard(sender, recipient, amount);

}

**Recommendation:**

if (\_isExcluded[sender] && !\_isExcluded[recipient]) {

\_transferFromExcluded(sender, recipient, amount);

} else if (!\_isExcluded[sender] && \_isExcluded[recipient]) {

\_transferToExcluded(sender, recipient, amount);

} else if (\_isExcluded[sender] && \_isExcluded[recipient]) {

\_transferBothExcluded(sender, recipient, amount);

} else {

\_transferStandard(sender, recipient, amount);

}

**Status**: **Open**

### **[INF3] Less meaningful variable:**

**MAX** variable names do not provide a clear picture of their purpose

**Recommendation: MAX** should be renamed to **MAX\_INT\_256**

**Status**: **Open**

### **[INF4] Variable defined but never used**

**‘contractTokenBalance’** defined but never used

**Recommendation:** Remove unused variables

**Status**: **Open**

### **[INF5] Public methods only being used externally**

‘public’ functions that are not inherited by the contract should be declared ‘external’ to save gas.

**Recommendation:** Make these methods external -

*name()*

*symbol()*

*decimals()*

*totalSupply()*

*balanceOf()*

*transfer()*

*allowance()*

*approve()*

*transferFrom()*

*increaseAllowance()*

*decreaseAllowance()*

*isExcludedFromReward()*

*totalFees()*

*deliver()*

*reflectionFromToken()*

*tokenFromReflection()*

*excludeFromReward()*

*excludeFromFee()*

*includeInFee()*

*isExcludedFromFee()*

**Status**: **Open**

### **[INF6] Constant calculations in the contract**

Precalculated initialization will save 2847 units of gas in deployment

uint256 private \_tTotal = 500\*\*1 \* 10\*\*6 \* 10\*\*18;

uint256 private \_rTotal = (MAX - (MAX % \_tTotal));

**Recommendation:** Replace the initialization as -

uint256 private \_tTotal = 500000000000000000000000000;

uint256 private \_rTotal = 115792089237316195423570985008687907853269984665640500000000000000000000000000;

10\*\*2 should be a precalculated constant and then used in **calculateTaxFee**.

**Status: Open**

### **[INF7] Use a constructor to set addresses**

The address is set in the contract should be provided dynamically via constructor on deployment for readability and correctness

**Recommendation:** Pass the address of **“walletZakat”** as a parameter of the constructor and set the value

**Status: Open**

### **[INF8] Getters should be at the bottom of the contract**

**Status: Open**

### **[INF9] Length calculation within the loop**

Reading from the state and fetching its length is a costly operation and doing such within a loop should be avoided

**Recommendation: \_excluded.length** should be calculated and stored in a variable before using in the for loop inside **`\_getCurrentSupply`** and **`includeInReward`** method.

**Status: Open**

### **[INF10] Inconsistent error messages**

The require checks in the contract have error messages containing the contract name in some places and the same has been skipped in most cases.

**Recommendation:** The error message should follow the same pattern and we recommend using the contract name along with the error message at all places  
Eg.

require(\_isExcluded[account], "MuslimCoins: Account is already excluded");

**Status: Open**

### **[INF11] Naming Conventions**

The contract follows a consistent naming convention of private variables with leading “\_” and public variables without it.

But we have missed complying with the condition for certain variable names -

**"calculateTaxFee"**, **"removeAllFee"** and **“restoreAllFee”** which are private.

**Recommendation:**

Add “\_” in private method names

**Status: Open**

### **[INF12] Presence of not implemented payable method**

The method **`receive()`** is not implemented but is present in the contract.

**Recommendation:** Remove the method from the contract or call the **`revert()`** method inside this **`receive()`**  method.

Eg.

receive() external payable {

revert("NOT\_IMPLEMENTED");

}

# **Binance Testnet Test Contract**

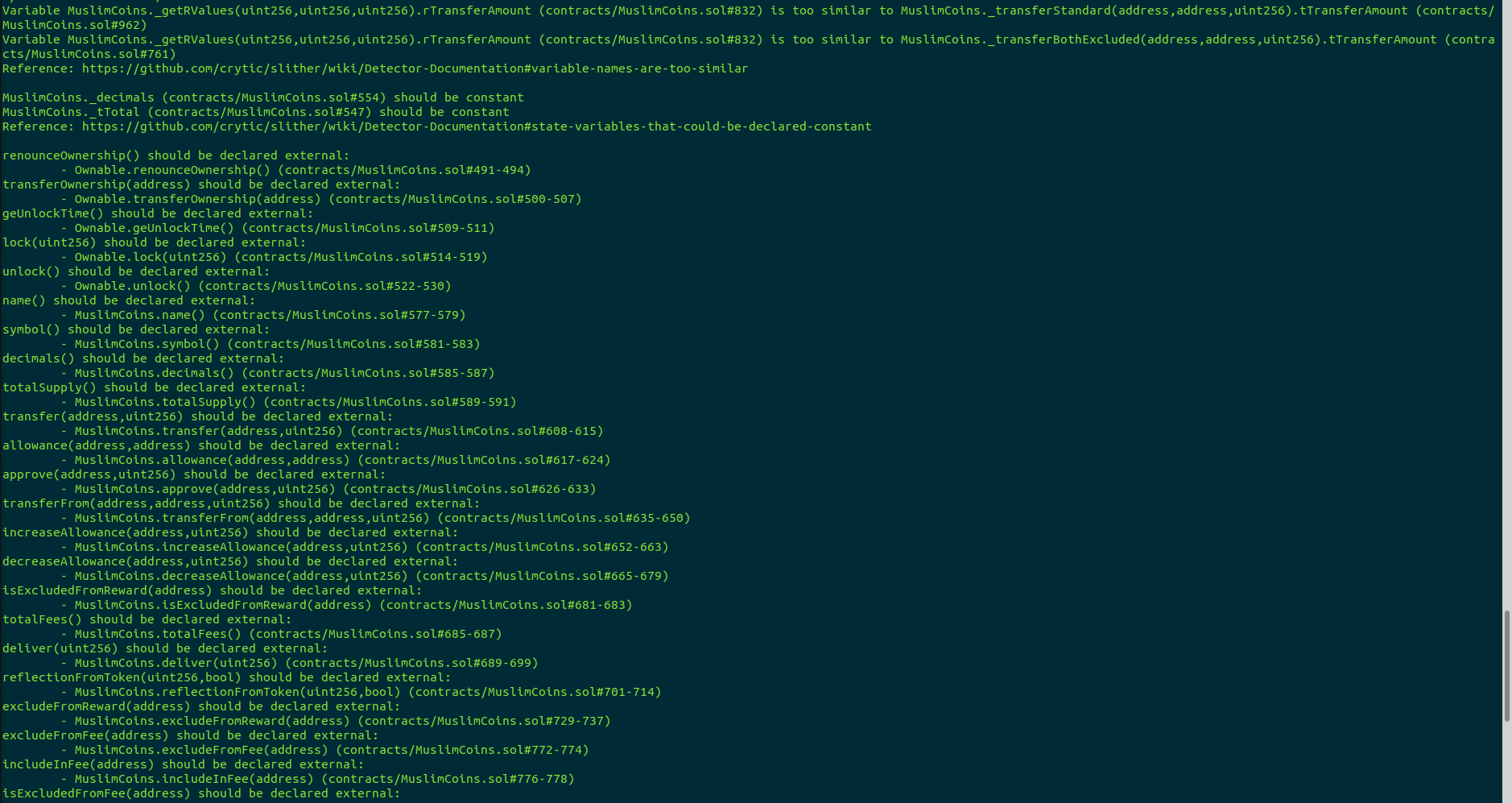
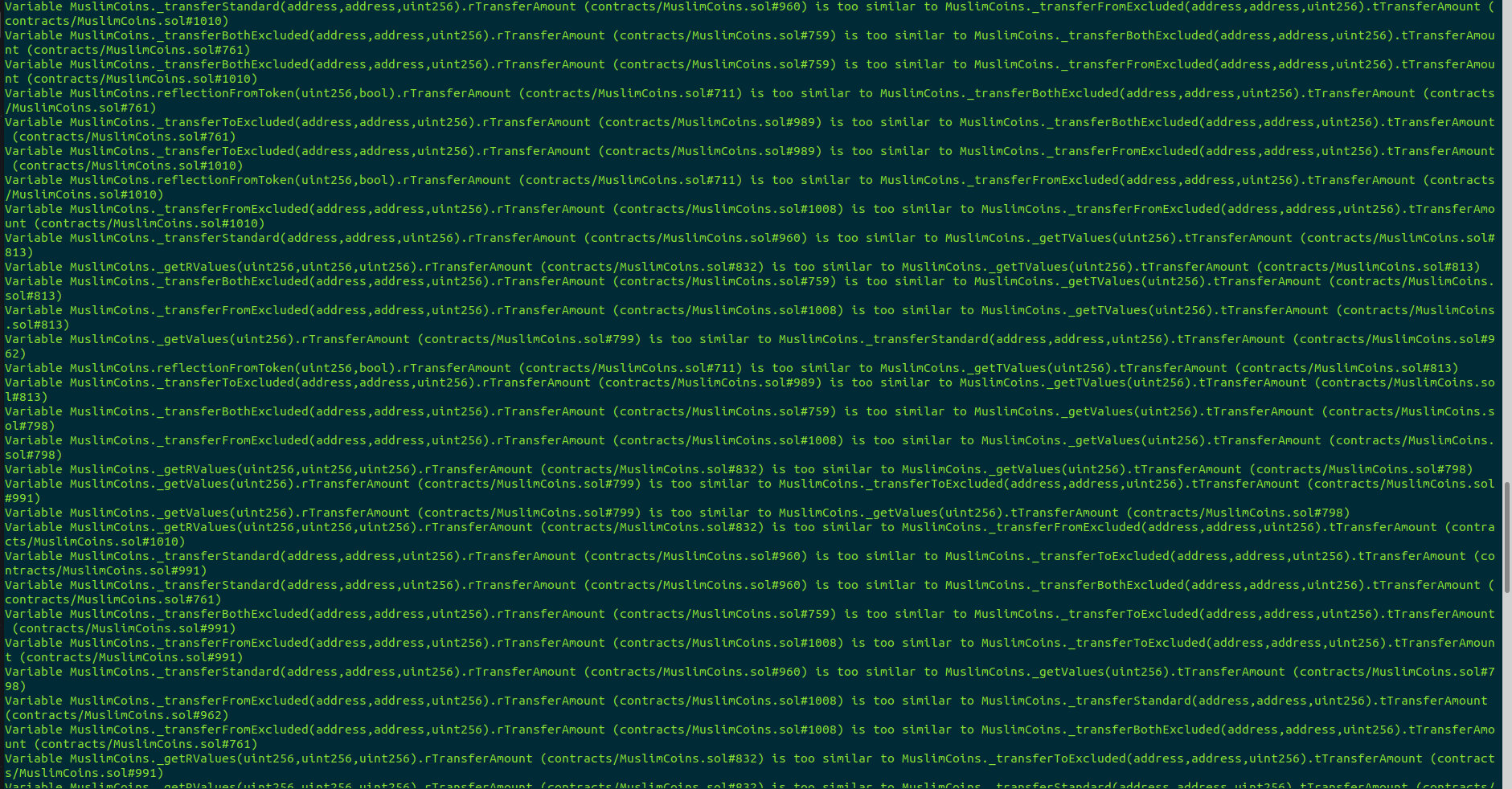
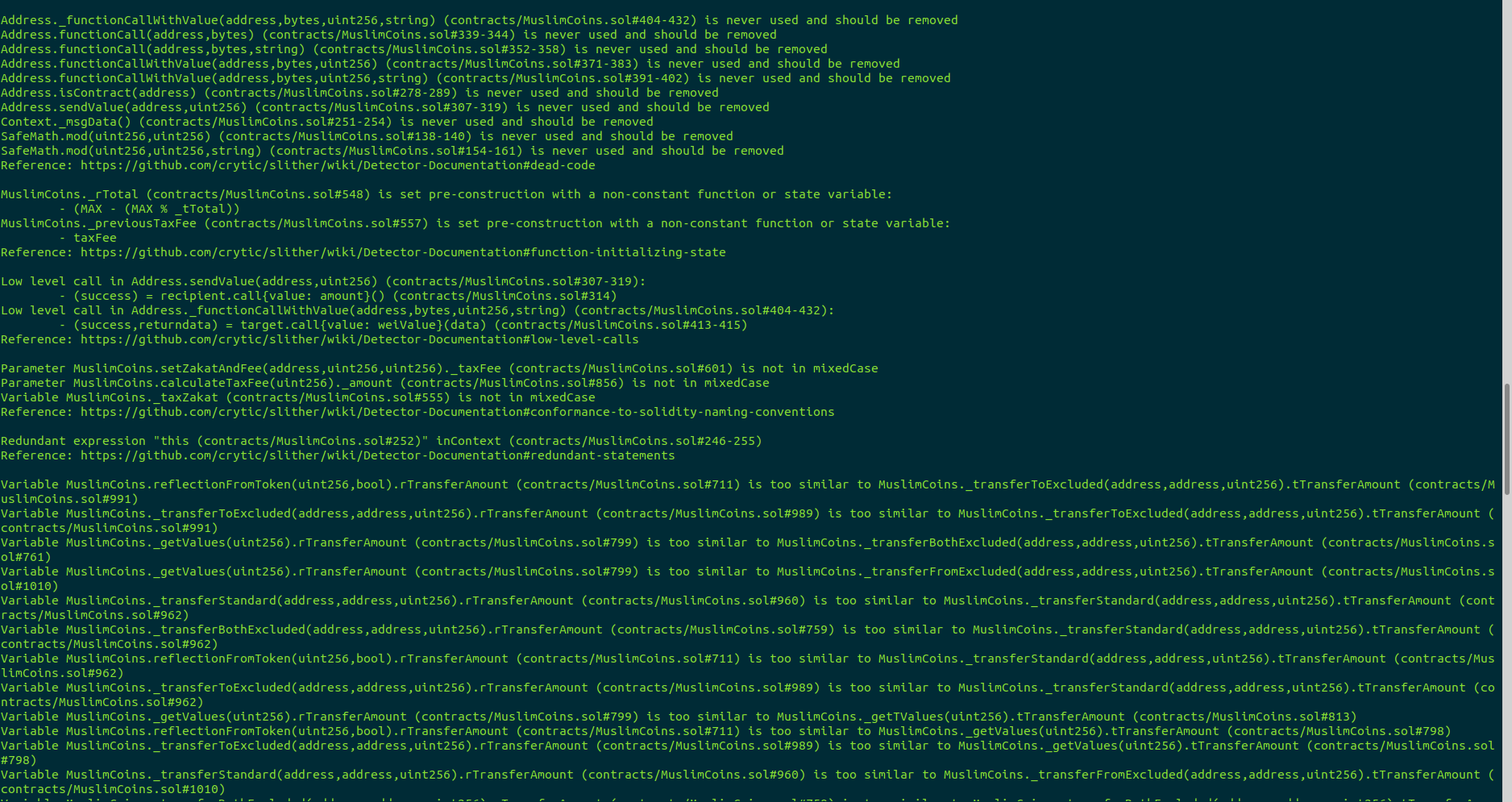
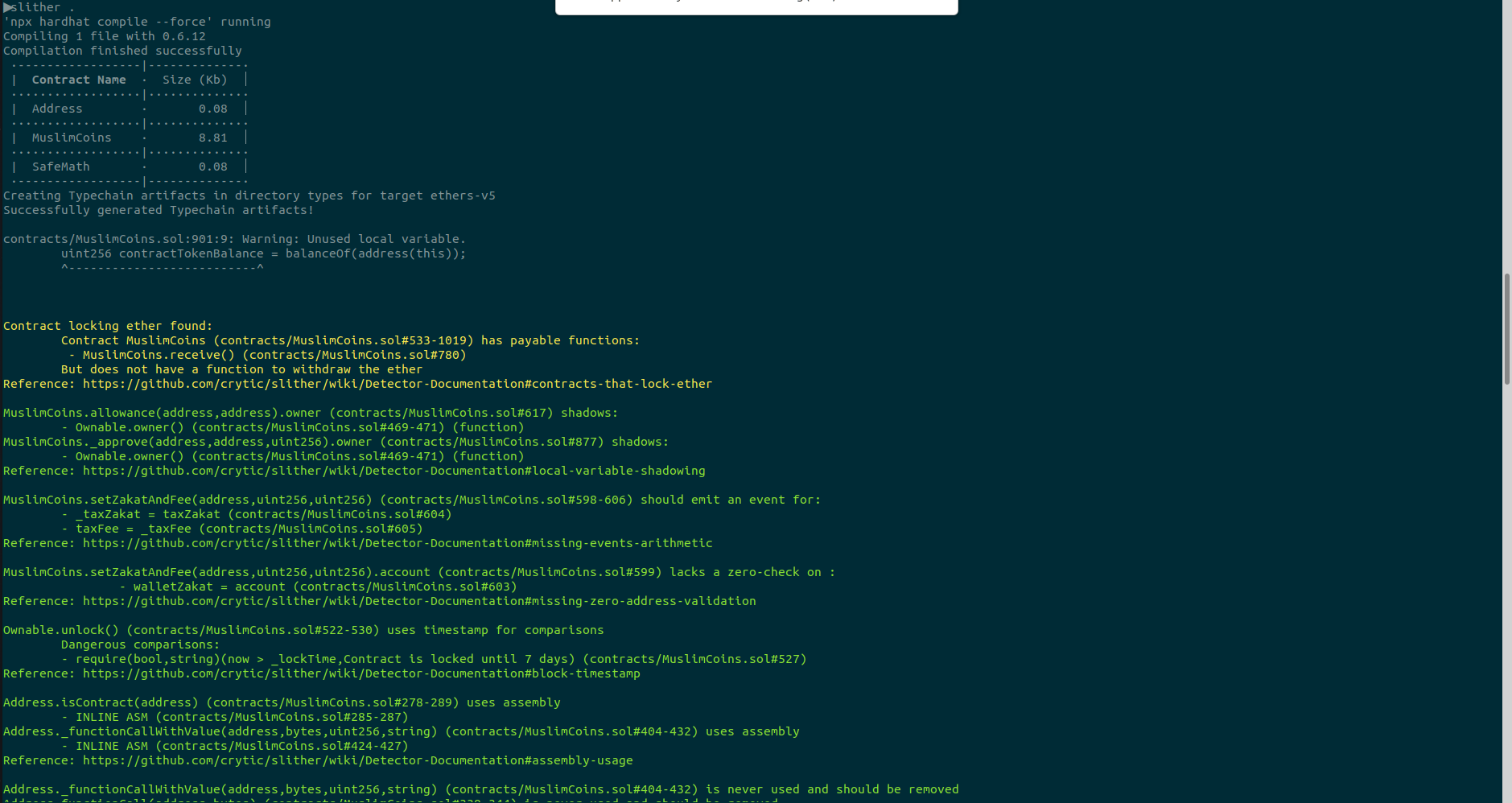
**MuslimCoins:** [0x23Ab9286aF3713783d994d675b1421824396Aeef](https://testnet.bscscan.com/address/0x23Ab9286aF3713783d994d675b1421824396Aeef#writeContract)

# Functional Tests

* [Deliver](https://testnet.bscscan.com/tx/0xd3a480f8dbfdfcd466c50bf48545499f1d7cda489ecf982866e1bded5256f1bf)
* [Exclude from fee](https://testnet.bscscan.com/tx/0xcb933a32c1ae06ce9a7f3cc60cc0e3c5605ed998c662812624e691766876424e)
* [Exclude from reward](https://testnet.bscscan.com/tx/0x0cb7a1481a4217eaa8d77b54e6343d91195d6eb312004fcbf369a0d5b33af016)
* [Include in fee](https://testnet.bscscan.com/tx/0x54f30a955068adb52cbba15777600146e24656407cf7fbcf0de859dcea85823b)
* [Include in reward](https://testnet.bscscan.com/tx/0xb7c9724ba752b69e1471e4e3b321d3213dc752472828e909af6de85a644c0eee)
* [Set zakat and fee](https://testnet.bscscan.com/tx/0x79df108c6d5ec2274381f4c6a4b6c440d7aa3267277cf43752aa37c39d852435)

# Automated Tests

**Slither:**

****

### **Results:**

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

# **Closing Summary**

Overall, smart contracts are very well written, documented, and adhere to guidelines. Several issues of Low severity and information issues have been reported.  
We recommend working on suggestions that are reported in order to improve the code quality of smart contracts.

## **Disclaimer**

Quillhash audit is not a security warranty, investment advice, or endorsement of the **MuslimCoins platform**. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the MuslimCoins Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.