



Smart Contract Audit

FOR

DMScoin

DATED : 27 Feb, 2024



AUDIT SUMMARY

Project name – DMScoin

Date: 27 Feb, 2024

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: **Passed**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	1	1
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1- Manual Review:

A line by line code review has been performed by audit ace team.

2- BSC Test Network: All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

3- Slither :

The code has undergone static analysis using Slither.

Testnet version:

The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:

<https://testnet.bscscan.com/address/0x04bcc12E356e1eFb959FEa4a75dD2a1C62220B00#code>



Token Information

Token Name : DMScoin

Token Symbol: DMS

Decimals: 18

Token Supply: 100000000000

Network: BscScan

Token Type: BEP-20

Token Address:

0xD4121306B742E9d9623CB541CA5Bd5BF13F0E204

Checksum:

ke1c3a4fbb6e83e8393a57617b5a5B21

Owner:

0xB7Cd7175B8D423B10640756Ee547AcF666e82A26
(at time of writing the audit)

Deployer:

0xB7Cd7175B8D423B10640756Ee547AcF666e82A26



TOKEN OVERVIEW

Fees:

Buy Fee: 0-0%

Sell Fee: 0-0%

Transfer Fee: 0-0%

Fees Privilege: Owner

Ownership: Owned

Minting: No

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No



AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

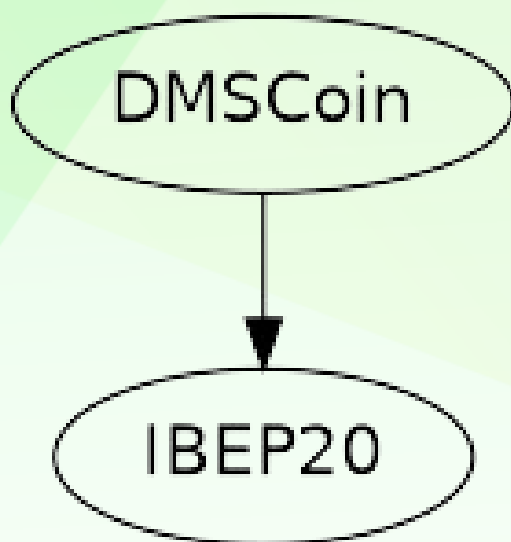
- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
 - Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
 - Test coverage analysis determines whether the test cases are covering the code and how much code is exercised when we run the test cases.
 - Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
 - Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.
-



VULNERABILITY CHECKLIST

- | | |
|------------------------------------|-------------------------------|
| ✓ Return values of low-level calls | ✓ Gasless Send |
| ✓ Private modifier | ✓ Using block.timestamp |
| ✓ Multiple Sends | ✓ Re-entrancy |
| ✓ Using Suicide | ✓ Tautology or contradiction |
| ✓ Gas Limitand Loops | ✓ Timestamp Dependence |
| ✓ Address hardcoded | ✓ Revert/require functions |
| ✓ Exception Disorder | ✓ Use of tx.origin |
| ✓ Using inline assembly | ✓ Integer overflow/underflow |
| ✓ Divide before multiply | ✓ Dangerous strict equalities |
| ✓ Missing Zero Address Validation | ✓ Using SHA3 |
| ✓ Compiler version not fixed | ✓ Using throw |
-

INHERITANCE TREE





STATIC ANALYSIS

A static analysis of the code was performed using Slither.
No issues were found.

```
INFO:Detectors:
Pragma version^0.8.0 (DMSCoin.sol#6) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter DMSCoin.allowance(address,address)._owner (DMSCoin.sol#56) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
DMSCoin.decimals (DMSCoin.sol#23) should be constant
DMSCoin.name (DMSCoin.sol#21) should be constant
DMSCoin.symbol (DMSCoin.sol#22) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
DMSCoin.owner (DMSCoin.sol#28) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:DMSCoin.sol analyzed (2 contracts with 93 detectors), 7 result(s) found
```



FUNCTIONAL TESTING

1- Approve (passed):

<https://testnet.bscscan.com/tx/0x820347531ab35b8ec6e0e4104e9493318f5e82b4bed4ca4988fd36f06a1013e3>

2- Burn (passed):

<https://testnet.bscscan.com/tx/0x77d28c61c1c4955c756f7edc17159a93bfd677ac04272a8bd3ba8e2f2d26cd04>

3- Transfer (passed):

<https://testnet.bscscan.com/tx/0x706f9c1f91ba410dc77a90c85f68019b64ce39dba1da89baaeaa161d2f43d3e4>



CLASSIFICATION OF RISK

Severity

Description

◆ Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
◆ High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
◆ Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
◆ Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
◆ Gas Optimization / Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity

Found

◆ Critical	0
◆ High-Risk	0
◆ Medium-Risk	0
◆ Low-Risk	1
◆ Gas Optimization / Suggestions	1

MANUAL TESTING

Centralization – Missing Visibility

Severity: Low

Subject: Visibility

Status: Open

Overview:

It's simply saying that no visibility was specified, so it's going with the default. This has been related to security issues in contracts.

```
mapping(address => uint256) balances;  
mapping(address => mapping(address => uint256)) allowed;
```

Suggestion:

You can easily silence the warning by adding the public/private.



MANUAL TESTING

Optimization

Severity: Informational

Subject: Floating Pragma

Status: Open

Overview:

It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

```
pragma solidity ^0.8.0;
```

Suggestion:

Adding the latest constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.



DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.



ABOUT AUDITACE

We specialize in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.



<https://auditace.tech/>



https://t.me/Audit_Ace



https://twitter.com/auditace_



<https://github.com/Audit-Ace>
