AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

DOGO MOON GO

DATED : 30 May 24'

AUDIT SUMMARY

Project name – DOGO MOON GO

Date: 30 May 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	3
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/0x9a9d7f5fa9635243cda8fa3a8000937a09d9c024#code

Token Information

Token Address:
0xD435b2ebBEA95A96f4d53E98BF8815aa8eee9B2c

Name: DOGO MOON GO

Symbol: DGM

Decimals: 9

Network: BscScan

Token Type: BEP-20

Owner: 0x0068D28c589783788090Ff94bc27949F0c3985e4

Deployer: --

Token Supply: 100000000

Checksum: Ae032c616934aeb47e6039f76b20d213

Testnet:
https://testnet.bscscan.com/address/0x9a9d7f5fa9635243cda8fa3a8000937a09d9c024#code

TOKEN OVERVIEW

Buy Fee: 0-0%

Sell Fee: 0-0%

Transfer Fee: 0-0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: 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 isexercised 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

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

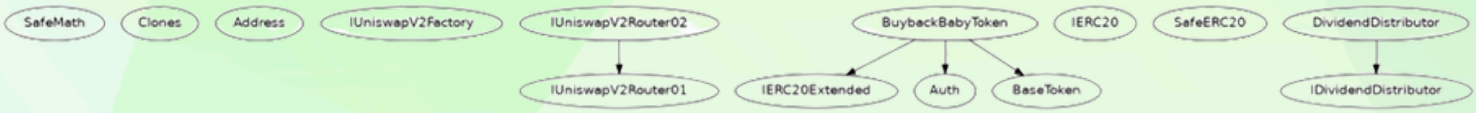
CLASSIFICATION OF RISK

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	3

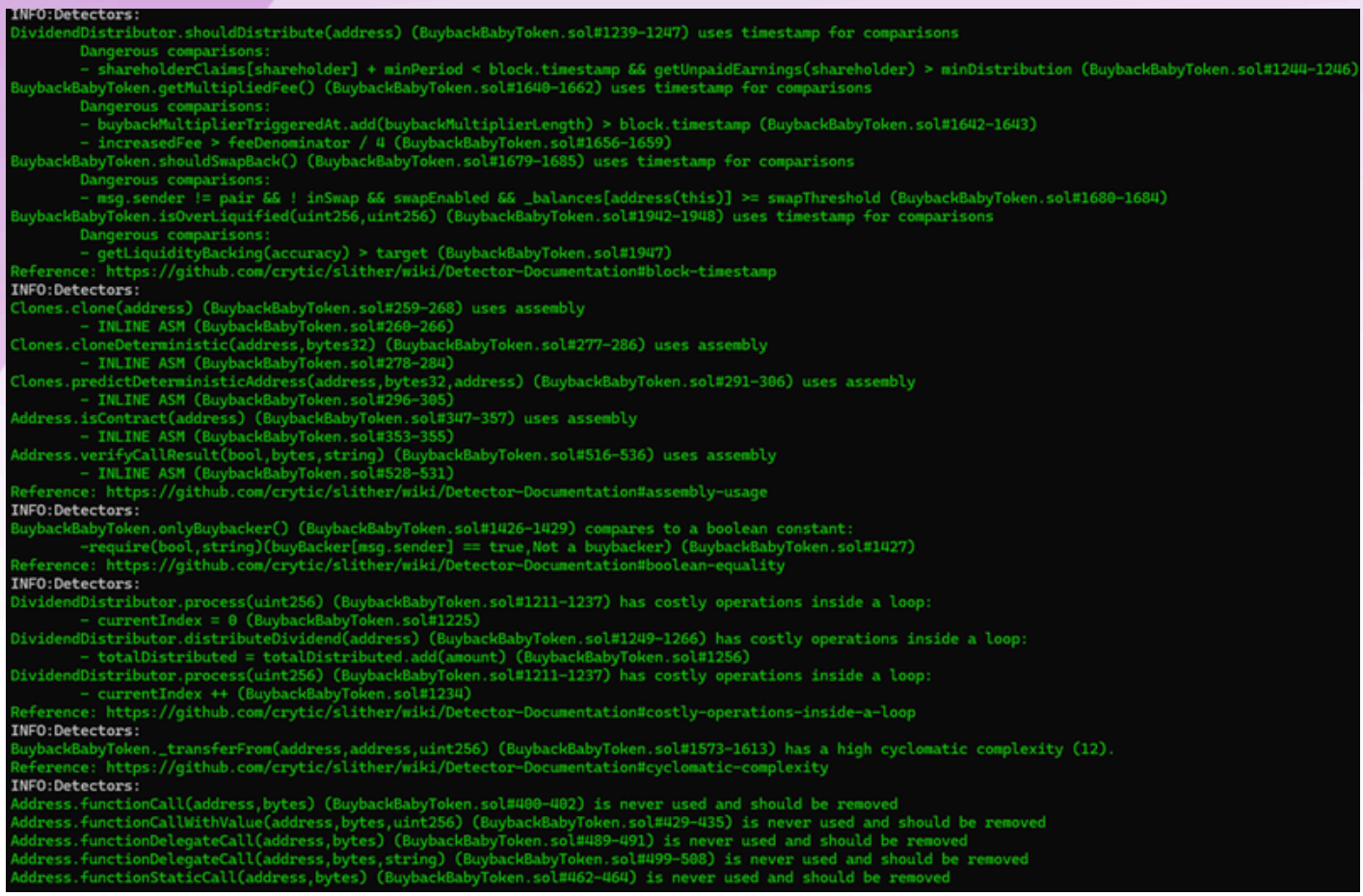
INHERITANCE TREE



POINTS TO NOTE

- The owner can transfer ownership.
- The owner can authorize/unauthorize other addresses.
- The owner can set distribution criteria for dividends, minimum period, and minimum distribution value.
- The owner can trigger buyback and clear buyback multiplier.
- The owner sets buyback status for a particular account.
- The owner set the fee receiver address.
- The owner can include/exclude wallets from fees, and dividends.
- The owner swap back the status and amount.

STATIC ANALYSIS



```
INFO:Detectors:
DividendDistributor.shouldDistribute(address) (BuybackBabyToken.sol#1239-1247) uses timestamp for comparisons
        Dangerous comparisons:
        - shareholderClaims[shareholder] + minPeriod < block.timestamp && getUnpaidEarnings(shareholder) > minDistribution (BuybackBabyToken.sol#1244-1246)
BuybackBabyToken.getMultipliedFee() (BuybackBabyToken.sol#1640-1662) uses timestamp for comparisons
        Dangerous comparisons:
        - buybackMultiplierTriggeredAt.add(buybackMultiplierLength) > block.timestamp (BuybackBabyToken.sol#1642-1643)
        - increasedFee > feeDenominator / 4 (BuybackBabyToken.sol#1656-1659)
BuybackBabyToken.shouldSwapBack() (BuybackBabyToken.sol#1679-1685) uses timestamp for comparisons
        Dangerous comparisons:
        - msg.sender != pair && ! inSwap && swapEnabled && _balances[address(this)] >= swapThreshold (BuybackBabyToken.sol#1680-1684)
BuybackBabyToken.isOverLiquified(uint256,uint256) (BuybackBabyToken.sol#1942-1948) uses timestamp for comparisons
        Dangerous comparisons:
        - getLiquidityBacking(accuracy) > target (BuybackBabyToken.sol#1947)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Clones.clone(address) (BuybackBabyToken.sol#259-268) uses assembly
        - INLINE ASM (BuybackBabyToken.sol#260-266)
Clones.cloneDeterministic(address,bytes32) (BuybackBabyToken.sol#277-286) uses assembly
        - INLINE ASM (BuybackBabyToken.sol#278-284)
Clones.predictDeterministicAddress(address,bytes32,address) (BuybackBabyToken.sol#291-306) uses assembly
        - INLINE ASM (BuybackBabyToken.sol#296-305)
Address.isContract(address) (BuybackBabyToken.sol#347-357) uses assembly
        - INLINE ASM (BuybackBabyToken.sol#353-355)
Address.verifyCallResult(bool,bytes,string) (BuybackBabyToken.sol#516-536) uses assembly
        - INLINE ASM (BuybackBabyToken.sol#528-531)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
BuybackBabyToken.onlyBuybacker() (BuybackBabyToken.sol#1426-1429) compares to a boolean constant:
        -require(bool,string)(buyBacker[msg.sender] == true,Not a buybacker) (BuybackBabyToken.sol#1427)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
DividendDistributor.process(uint256) (BuybackBabyToken.sol#1211-1237) has costly operations inside a loop:
        - currentIndex = 0 (BuybackBabyToken.sol#1225)
DividendDistributor.distributeDividend(address) (BuybackBabyToken.sol#1249-1266) has costly operations inside a loop:
        - totalDistributed = totalDistributed.add(amount) (BuybackBabyToken.sol#1256)
DividendDistributor.process(uint256) (BuybackBabyToken.sol#1211-1237) has costly operations inside a loop:
        - currentIndex ++ (BuybackBabyToken.sol#1234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
INFO:Detectors:
BuybackBabyToken._transferFrom(address,address,uint256) (BuybackBabyToken.sol#1573-1613) has a high cyclomatic complexity (12).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
Address.functionCall(address,bytes) (BuybackBabyToken.sol#400-402) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (BuybackBabyToken.sol#429-435) is never used and should be removed
Address.functionDelegateCall(address,bytes) (BuybackBabyToken.sol#489-491) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (BuybackBabyToken.sol#499-508) is never used and should be removed
Address.functionStaticCall(address,bytes) (BuybackBabyToken.sol#462-464) is never used and should be removed
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

FUNCTIONAL TESTING

1- Approve (passed):

https://testnet.bscscan.com/tx/0xf032cfdf07d0c0fa0d7d61363cce8befd967d5f9a41b13f436063e8e676c7a45

2- Set Fees (passed):

https://testnet.bscscan.com/tx/0x24f5b3af651d0e2943848fecc983cee8161ca481934e20debb5f6f99be2c4bf8

MANUAL TESTING

Centralization – Missing Events

Severity: Low

Subject: Missing Events

Status: Open

Overview:

They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function setDistributionCriteria(
    uint256 _minPeriod,
    uint256 _minDistribution
  ) external onlyOwner {
    distributor.setDistributionCriteria(_minPeriod, _minDistribution);
  }
function setAutoBuybackSettings(
    bool _enabled,
    uint256 _cap,
    uint256 _amount,
    uint256 _period
  ) external authorized {
    require(_period > 0, "Period must be greater than 0");
    autoBuybackEnabled = _enabled;
    autoBuybackCap = _cap;
    autoBuybackAccumulator = 0;
    autoBuybackAmount = _amount;
    autoBuybackBlockPeriod = _period;
    autoBuybackBlockLast = block.number;
  }
function setBuybackMultiplierSettings(
    uint256 numerator,
    uint256 denominator,
    uint256 length
```

MANUAL TESTING

```
) external authorized {
    require(length <= 2 hours, "Length must be less than 2 hours");
    require(numerator / denominator <= 2 && numerator > denominator);
    buybackMultiplierNumerator = numerator;
    buybackMultiplierDenominator = denominator;
    buybackMultiplierLength = length;
  }
function setSwapBackSettings(bool _enabled, uint256 _amount)
    external
    authorized
  {
    require(
      _enabled && _amount >= _totalSupply / 100_000,
      "Swapback amount should be at least 0.001% of total supply"
    );
    swapEnabled = _enabled;
    swapThreshold = _amount;
  }
function setTargetLiquidity(uint256 _target, uint256 _denominator)
    external
    authorized
  {
    require(_denominator > 0, "Denominator must be greater than 0");
    targetLiquidity = _target;
    targetLiquidityDenominator = _denominator;
  }
```

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 pragma solidity ^0.8.4;

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.

MANUAL TESTING

Optimization

Severity: Informational

Subject: Remove Safe Math

Status: Open

Line: 21-232

Overview:

compiler version above 0.8.0 can control arithmetic overflow/underflow, it is recommended to remove the unwanted code to avoid high gas fees.

MANUAL TESTING

Optimization

Severity: Optimization

Subject: Remove unused code.

Status: Open

Overview:

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice. though to avoid them.

```
function clone(address implementation) internal returns (address
instance) {
    assembly {
      let ptr := mload(0x40)
      mstore(ptr,
0x3d602d80600a3d3981f3363d3d373d3d3d363d7300000000000000000000
00000000)
      mstore(add(ptr, 0x14), shl(0x60, implementation))
      mstore(add(ptr, 0x28),
0x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000
0000000)
instance := create(0, ptr, 0x37)
    }
    require(instance != address(0), "ERC1167: create failed");
  }
function cloneDeterministic(address implementation, bytes32 salt)
internal returns (address instance) {
    assembly {
      let ptr := mload(0x40)
      mstore(ptr,
0x3d602d80600a3d3981f3363d3d373d3d3d363d7300000000000000000000
00000000)
      mstore(add(ptr, 0x14), shl(0x60, implementation))
      mstore(add(ptr, 0x28),
```

MANUAL TESTING

```
0x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000000000)
instance := create2(0, ptr, 0x37, salt)
    }
    require(instance != address(0), "ERC1167: create2 failed");
  }
 function predictDeterministicAddress(address implementation, bytes32 salt)
    internal
    view
    returns (address predicted)
  {
    return predictDeterministicAddress(implementation, salt, address(this));
  }
}
function sendValue(address payable recipient, uint256 amount) internal {
    require(address(this).balance >= amount, "Address: insufficient balance");

    (bool success, ) = recipient.call{value: amount}("");
    require(success, "Address: unable to send value, recipient may have reverted");
  }
function functionCall(address target, bytes memory data) internal returns (bytes memory) {
    return functionCall(target, data, "Address: low-level call failed");
  }
function functionCallWithValue(
    address target,
    bytes memory data,
    uint256 value
  ) internal returns (bytes memory) {
    return functionCallWithValue(target, data, value, "Address: low-level call with value failed");
```

MANUAL TESTING

```
  }
function functionStaticCall(address target, bytes memory data) internal
view returns (bytes memory) {
    return functionStaticCall(target, data, "Address: low-level static call
failed");
  }
```

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 specializes 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