



MissDoge

Smart Contract Security Audit

Audit Report
Aug, 2021



<http://tokenaudit.net/>



<https://t.me/TokenAuditt>

Table of contents

Project Introduction	2
Auditing Methodologies applied:	3
Static Analysis	4
Auditing Tools	4
Tokenomics:	5
Issues Checking	6
Severity Issue Categories	8
Issues Found	9
Low level severity issues	9
Automated Testing	12
Sol-Hint Tool:	15
Functional View	16
Inheritance Chart	17
Audit Findings Results	18

Project Introduction

Here's the magnificent Miss of Baby Doge 🐶 seeing her family members being famous, she couldn't bear with that and she knew this is the time for her show.

Name	MISSDOGE (MDOGE)
Total Supply	1,000,000,000,000,000
Type	BSC Token
Platform	Ethereum / Solidity
Holders	19,829 addresses
Deployed Contract	0xA72Ff2B799324B042AE379809eE54dACE99db3A5

Token Audit Team performed a security audit for **MISSDOGE** smart contracts during the period of Aug 15, 2021 to Aug 16, 2021.

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

<https://github.com/MissDoge/contract/blob/main/missdoge.sol>

Auditing Methodologies applied:

- In this audit, we consider the following crucial features of the code.
- The overall quality of code.
- Whether the implementation of ERC 20 standards.
- Whether the code is secure.
- Gas Optimization
- Code is safe from reentrancy and other vulnerabilities

Manual Audit

- Manually analyzing the source code line-by-line in an attempt to identify security vulnerabilities.
- Gas Consumption and optimization
- Assessing the overall project structure, complexity & quality.
- Checking whether all the libraries used in the code of the latest version.

Automated Audit

- Projects can be Automated using these tools with Slither, Manticore, SolGraph others.
 - Performing Unit testing.
 - Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
- Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

Static Analysis

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

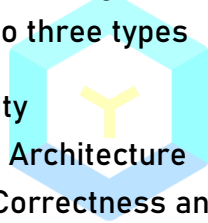
Auditing Tools

Language: Solidity

Platform and tools: Slither, Manti-Core, VScode, Solhint, Solc-select, Solidity-coverage

Audit Aim: The focus of the audit was to verify whether the smart contract is secure, resilient, and working according to the standard specs. The audit activity can be categorized into three types

- Security
- Sound Architecture
- Code Correctness and Quality



TOKEN AUDIT

Tokenomics:

The tokenomics were as follows (in number of tokens):

1,000,000,000,000,000 Total Supply

6% of token will be buy back

1% of token will be redistributed equally to all holders as a reward.

4% token will be for marketting (Influencer marketing push)

This wallet exists to ensure the ongoing of the project and consists of the following parts:



TOKEN AUDIT

1% Airdrop

3% Burn

50% Pre-sale

35% Liquidity

11% DxSale Pre-sale Fee

Issues Checking

We have scanned this smart contract code for commonly known and more specific Vulnerabilities that are below listed:

SN	Issue Description	Status
1	Re-entrancy	Verified
2	Compiler errors	Verified
3	Timestamp Dependence	Verified
4	Unsafe external calls	Verified
5	Gas Limit and Loops	Verified
6	DoS with Block Gas Limit	Verified
7	Private user data leaks	Verified
8	Code clones, functionality duplication	Verified
9	Style guide violation	Verified
10	Costly Loop	Verified
11	Balance equality	Verified
12	Unchecked math	Verified

13	Integer overflow/underflow	Verified
14	Cross-function Race Condition	Verified
15	Fallback function security	Verified
16	Data Consistency	Verified
17	Balance equality	Verified
18	ERC20 API violation	Verified
19	Deployment Consistency	Verified
20	Arithmetic accuracy	Verified
21	Transaction-Ordering Dependence (TOD) / Front Running	Verified
22	Address hardcoded	Verified
23	Scoping and Declarations	Verified
24	Implicit visibility level	Verified
25	Call Depth Attack (deprecated)	Verified

Severity Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	The issue affects the ability of the contract to compile or operate in a significant way.
Medium	Issues on this level could potentially bring problems and should eventually be fixed.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution.
Informational	The issue has no impact on the contract's ability to operate.

Issues Found

- **Critical severity issues** - No Critical severity issues found.
- **High severity issues** - 0 Critical severity issues found.
- **Medium severity issues** - 0 medium severity issues found.
- **Low severity issues** - 3 low severity issues were found.
- **Informational** - 1 Informational severity issues were found.

High	Medium	Low	Informational
0	0	3	1

Low level severity issues

1. Missing Range Check for Input Variable

Description

The owner can set the following state variables arbitrary large or small causing potential risks in fees and anti whale:

```
_buyBackMaxTimeForHistories  
_buyBackDivisor  
_buyBackTimeInterval  
_intervalMinutesForSwap  
_taxFee
```

Suggestion:

We recommend setting ranges and check the above input variables.

2. Missing Events for Significant Transactions

```
pragma solidity ^0.8.4;
```

Description

Contracts should be deployed using the same compiler version/flags with which they have been tested. Locking the pragma (for e.g., by not using ^ in pragma solidity 0.8.0) ensures that contracts do not accidentally get deployed using an older compiler version with unfixed bugs.

Remediation:

Lock the pragma version.



TOKEN AUDIT

3. Solidity naming conventions

Severity: Informational

Description:

In the contract, many function names were found to be starting with capital letters. Functions other than constructors should use mixedCase.

Examples: getBalance, transfer, verifyOwner

Remediation:

Solidity naming convention

Informational level severity issues

1. Public function that could be declared external

Severity: Informational

Description:

The following public functions that are never called by the contract should be declared external to save gas:

- totalFees()
- deliver()
- reflectionFromToken()
- tokenFromReflection()
- excludeFromReward()

Remediation:

Use the external attribute for functions never called from the contract

Automated Testing

We have to use Automated testing Slither. It is an Automated Analysis Tool in Smart Contract.

```
isExcludedFromReward(address) should be declared external:
- MissDoge.isExcludedFromReward(address) (MissDoge.sol#561-563)
totalFees() should be declared external:
- MissDoge.totalFees() (MissDoge.sol#565-567)
minimumTokensBeforeSwapAmount() should be declared external:
- MissDoge.minimumTokensBeforeSwapAmount() (MissDoge.sol#569-571)
buyBackUpperLimitAmount() should be declared external:
- MissDoge.buyBackUpperLimitAmount() (MissDoge.sol#573-575)
deliver(uint256) should be declared external:
- MissDoge.deliver(uint256) (MissDoge.sol#577-584)
reflectionFromToken(uint256,bool) should be declared external:
- MissDoge.reflectionFromToken(uint256,bool) (MissDoge.sol#587-596)
excludeFromReward(address) should be declared external:
- MissDoge.excludeFromReward(address) (MissDoge.sol#604-612)
isExcludedFromFee(address) should be declared external:
- MissDoge.isExcludedFromFee(address) (MissDoge.sol#881-883)
excludeFromFee(address) should be declared external:
- MissDoge.excludeFromFee(address) (MissDoge.sol#885-887)
includeInFee(address) should be declared external:
- MissDoge.includeInFee(address) (MissDoge.sol#889-891)
setBuyBackEnabled(bool) should be declared external:
- MissDoge.setBuyBackEnabled(bool) (MissDoge.sol#926-929)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither: ./MissDoge.sol analyzed (10 contracts with 75 detectors), 132 result(s) found
```

```
External calls sending eth:
- _transfer(sender,recipient,amount) (MissDoge.sol#546)
- recipient.transfer(amount) (MissDoge.sol#946)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens(value: amount){0,path,deadAddress,block.timestamp.add(300)} (MissDoge.sol#720-725)
State variables written after the call(s):
- _approve(sender,msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (MissDoge.sol#547)
- _allowances[owner][spender] = amount (MissDoge.sol#631)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in MissDoge. transfer(address,address,uint256) (MissDoge.sol#635-673):
External calls:
- swapTokens(contractTokenBalance) (MissDoge.sol#653)
- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (MissDoge.sol#702-708)
- buyBackTokens(balance.div(100)) (MissDoge.sol#661)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens(value: amount){0,path,deadAddress,block.timestamp.add(300)} (MissDoge.sol#720-725)
External calls sending eth:
- swapTokens(contractTokenBalance) (MissDoge.sol#653)
- recipient.transfer(amount) (MissDoge.sol#946)
- buyBackTokens(balance.div(100)) (MissDoge.sol#661)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens(value: amount){0,path,deadAddress,block.timestamp.add(300)} (MissDoge.sol#720-725)
Event emitted after the call(s):
- SwapETHForTokens(amount,path) (MissDoge.sol#727)
- buyBackTokens(balance.div(100)) (MissDoge.sol#661)
- Transfer(sender,recipient,tTransferAmount) (MissDoge.sol#769)
- tokenTransfer(from,to,amount,takeFee) (MissDoge.sol#672)
- Transfer(sender,recipient,tTransferAmount) (MissDoge.sol#789)
- tokenTransfer(from,to,amount,takeFee) (MissDoge.sol#672)
- Transfer(sender,recipient,tTransferAmount) (MissDoge.sol#779)
- tokenTransfer(from,to,amount,takeFee) (MissDoge.sol#672)
- Transfer(sender,recipient,tTransferAmount) (MissDoge.sol#800)
- tokenTransfer(from,to,amount,takeFee) (MissDoge.sol#672)
Reentrancy in MissDoge.constructor() (MissDoge.sol#489-508):
External calls:
```

```

00)) (MissDoge.sol#720-725)
    Event emitted after the call(s):
    - Approval(owner, spender, amount) (MissDoge.sol#632)
      - _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, ERC20: transfer amount exceeds allowance)) (Miss
Doge.sol#547)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Ownable.unlock() (MissDoge.sol#200-205) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool, string)(block.timestamp > lockTime, Contract is locked until 7 days) (MissDoge.sol#202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (MissDoge.sol#95-104) uses assembly
    - INLINE ASM (MissDoge.sol#102)
Address.functionCallWithValue(address, bytes, uint256, string) (MissDoge.sol#132-149) uses assembly
    - INLINE ASM (MissDoge.sol#141-144)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCallWithValue(address, bytes, uint256, string) (MissDoge.sol#132-149) is never used and should be removed
Address.functionCall(address, bytes) (MissDoge.sol#115-117) is never used and should be removed
Address.functionCall(address, bytes, string) (MissDoge.sol#119-121) is never used and should be removed
Address.functionCallWithValue(address, bytes, uint256) (MissDoge.sol#123-125) is never used and should be removed
Address.functionCallWithValue(address, bytes, uint256, string) (MissDoge.sol#127-130) is never used and should be removed
Address.isContract(address) (MissDoge.sol#95-104) is never used and should be removed
Address.sendValue(address, uint256) (MissDoge.sol#106-112) is never used and should be removed
Context._msgData() (MissDoge.sol#18-21) is never used and should be removed
MissDoge.addLiquidity(uint256, uint256) (MissDoge.sol#730-743) is never used and should be removed
SafeMath.mod(uint256, uint256) (MissDoge.sol#83-85) is never used and should be removed
SafeMath.mod(uint256, uint256, string) (MissDoge.sol#87-90) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

```

```

INFO:Detectors:
MissDoge._rTotal (MissDoge.sol#436) is set pre-construction with a non-constant function or state variable:
    - (MAX - (MAX % _rTotal))
MissDoge._previousTaxFee (MissDoge.sol#445) is set pre-construction with a non-constant function or state variable:
    - _taxFee
MissDoge._previousLiquidityFee (MissDoge.sol#448) is set pre-construction with a non-constant function or state variable:
    - _liquidityFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Pragma version^0.8.4 (MissDoge.sol#11) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address, uint256) (MissDoge.sol#106-112):
    - (success) = recipient.call{value: amount}() (MissDoge.sol#110)
Low level call in Address.functionCallWithValue(address, bytes, uint256, string) (MissDoge.sol#132-149):
    - (success, returndata) = target.call{value: weiValue}(data) (MissDoge.sol#135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (MissDoge.sol#244) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (MissDoge.sol#245) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (MissDoge.sol#261) is not in mixedCase
Function IUniswapV2Router01.WETH() (MissDoge.sol#282) is not in mixedCase
Parameter MissDoge.calculateTaxFee(uint256). amount (MissDoge.sol#854) is not in mixedCase
Parameter MissDoge.calculateLiquidityFee(uint256). amount (MissDoge.sol#860) is not in mixedCase
Parameter MissDoge.setNumTokensSellToAddToLiquidity(uint256). minimumTokensBeforeSwap (MissDoge.sol#909) is not in mixedCase
Parameter MissDoge.setMarketingAddress(address). marketingAddress (MissDoge.sol#917) is not in mixedCase
Parameter MissDoge.setSwapAndLiquifyEnabled(bool). enabled (MissDoge.sol#921) is not in mixedCase
Parameter MissDoge.setBuyBackEnabled(bool). enabled (MissDoge.sol#926) is not in mixedCase
Variable MissDoge._taxFee (MissDoge.sol#444) is not in mixedCase
Variable MissDoge._liquidityFee (MissDoge.sol#447) is not in mixedCase
Variable MissDoge._maxTxAmount (MissDoge.sol#452) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (MissDoge.sol#19)" inContext (MissDoge.sol#13-22)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

```




```

isExcludedFromReward(address) should be declared external:
  - MissDoge.isExcludedFromReward(address) (MissDoge.sol#561-563)
totalFees() should be declared external:
  - MissDoge.totalFees() (MissDoge.sol#565-567)
minimumTokensBeforeSwapAmount() should be declared external:
  - MissDoge.minimumTokensBeforeSwapAmount() (MissDoge.sol#569-571)
buyBackUpperLimitAmount() should be declared external:
  - MissDoge.buyBackUpperLimitAmount() (MissDoge.sol#573-575)
deliver(uint256) should be declared external:
  - MissDoge.deliver(uint256) (MissDoge.sol#577-584)
reflectionFromToken(uint256,bool) should be declared external:
  - MissDoge.reflectionFromToken(uint256,bool) (MissDoge.sol#587-596)
excludeFromReward(address) should be declared external:
  - MissDoge.excludeFromReward(address) (MissDoge.sol#604-612)
isExcludedFromFee(address) should be declared external:
  - MissDoge.isExcludedFromFee(address) (MissDoge.sol#881-883)
excludeFromFee(address) should be declared external:
  - MissDoge.excludeFromFee(address) (MissDoge.sol#885-887)
includeInFee(address) should be declared external:
  - MissDoge.includeInFee(address) (MissDoge.sol#889-891)
setBuyBackEnabled(bool) should be declared external:
  - MissDoge.setBuyBackEnabled(bool) (MissDoge.sol#926-929)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:./MissDoge.sol analyzed (10 contracts with 75 detectors), 132 result(s) found

```

Sol-Hint Tool: TOKEN AUDIT

A linter for Solidity that provides both Security and Style Guide validations.

```
PS E:\token audit vs code>  missdodge.sol
```

```

missdodge.sol
19:2  error  Line length must be no more than 120 but current length is 132  max-line-length
119:2 error  Line length must be no more than 120 but current length is 122  max-line-length
127:2 error  Line length must be no more than 120 but current length is 146  max-line-length
132:2 error  Line length must be no more than 120 but current length is 149  max-line-length
547:2 error  Line length must be no more than 120 but current length is 130  max-line-length
557:2 error  Line length must be no more than 120 but current length is 138  max-line-length
764:2 error  Line length must be no more than 120 but current length is 146  max-line-length
773:2 error  Line length must be no more than 120 but current length is 146  max-line-length
783:2 error  Line length must be no more than 120 but current length is 146  max-line-length
793:2 error  Line length must be no more than 120 but current length is 146  max-line-length
821:2 error  Line length must be no more than 120 but current length is 147  max-line-length

```

✖11 problems (11 errors, 0 warnings)

Functional View

Function	Return Type	Test Result
name	Public	Verified
symbol	Public	Verified
decimals	Public	Verified
totalSupply	Public	Verified
balanceOf	Public	Verified
transfer	Public	Verified
allowance	Public	Verified
approve	Public	Verified
TransferFrom	Public	Verified
increaseAllowance	Public	Verified
decreaseAllowance	Public	Verified
isExcludedFromReward	Public	Verified
totalFees	Public	Verified
deliver	Public	Verified

reflectionFromToken	Public	Verified
tokenFromReflection	Public	Verified
excludeFromReward	Public	Verified
includeInReward	Public	Verified
transferBothExcluded	Public	Verified



Audit Findings Results

There were **3 Low** issues and **1 Informational** found during the audit. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner. None of the critical issues were resolved.

Generally, the contracts are well written and structured. The findings during the audit have some impact on contract performance or security

Disclaimer

This audit does not provide a security or correctness guarantee of audited smart contracts. You agree that your access and/or use, including but not limited to any services, products, platforms, content, will be at your Own risk. Smart contract remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security.



<http://tokenaudit.net/>



<https://t.me/TokenAudit>



<https://twitter.com/AuditToken>