MetaDoge

Smart Contract Audit Report

SYSFIXED

09 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
MetaDoge	MetaDoge	Binance Smart Chain

Addresses

Contract address	0xf3B185ab60128E4C08008Fd90C3F1F01f4B78d50
Contract deployer address	0x515C65E26BBF5dA731B5Ce6679b9f7fA0230B352

Project Website

https://t.me/Metadoge_GlobalCommunity

Codebase

https://bscscan.com/address/0xf3B185ab60128E4C08008Fd90C3F1F01f4B78d50#code

SUMMARY

Then MetaDoge was born in 2023, which also has the blessing of global traffic and the attention of major media around the world; Including the highlights of the global encryption world; 2023 Hardcore IP Opens the Best DAO Organization in the New Era. Return of the king and continue the legend.

Contract Summary

Documentation Quality

MetaDoge provides a very good documentation with standard of solidity base code.

- The technical description is provided clearly and structured and also dont have any high risk issue.

Code Quality

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by MetaDoge with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 588, 588, 619, 619, 670, 678, 720, 840, 840, 859, 859, 863, 863, 889, 936, 938, 1108, 1138, 1201, 1276, 1276, 1420, 1430, 1434, 1513, 1743, 1775, 1798, 1799, 1834, 1870, 1913, 1917, 1929, 1936, 1945 and 1513.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 6 and 1899.
- SWC-110 SWC-123 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 572, 573, 574, 577, 578, 579, 671, 679, 721, 968, 969, 990, 991, 992, 1426, 1484, 1514 and 1519.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 800 and 920.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 632 and 889.

CONCLUSION

We have audited the MetaDoge project released on January 2023 to discover issues and identify potential security vulnerabilities in MetaDoge Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

The security audit report provides a satisfactory result with some low-risk issues.

The issues found in the MetaDoge smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, a state variable visibility is not set, weak sources of randomness, tx.origin as a part of authorization control and out of bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value. For "tx.origin" issues we recommend to avoid this issue, using "tx.origin" as a security control that can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing. We recommend to don't using any of those environment variables as sources of randomness and being aware that the use of these variables introduces a certain level of trust into miners.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	PASS
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	ISSUE FOUND
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	ISSUE FOUND
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

Typographical Error	SWC-129	A typographical error can occur for example when the intent of a defined operation is to sum a number to a variable.	PASS
Override control character	SWC-130	Malicious actors can use the Right-To-Left-Override unicode character to force RTL text rendering and confuse users as to the real intent of a contract.	PASS
Unused variables	SWC-131 SWC-135	Unused variables are allowed in Solidity and they do not pose a direct security issue.	PASS
Unexpected Ether balance	SWC-132	Contracts can behave erroneously when they strictly assume a specific Ether balance.	PASS
Hash Collisions Variable	SWC-133	Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The transfer() and send() functions forward a fixed amount of 2300 gas.	PASS
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	PASS

SMART CONTRACT ANALYSIS

Started	Sunday Jan 08 2023 04:26:48 GMT+0000 (Coordinated Universal Time)
Finished	Monday Jan 09 2023 09:27:19 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Token.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 588

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
587
588   swapTokensAtAmount = __totalSupply * (10**14);
589
590   IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
//0x10ED43C718714eb63d5aA57B78B54704E256024E
591   // Create a uniswap pair for this new token
592
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 588

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
587
588   swapTokensAtAmount = __totalSupply * (10**14);
589
590   IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
//0x10ED43C718714eb63d5aA57B78B54704E256024E
591   // Create a uniswap pair for this new token
592
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 619

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
618   */
619   _mint(tokenReceiver, __totalSupply * (10**18));
620   }
621
622   function setSwapTokensAtAmount(uint256 newValue) public onlyOwner{
623
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 619

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
618   */
619   _mint(tokenReceiver, __totalSupply * (10**18));
620   }
621
622   function setSwapTokensAtAmount(uint256 newValue) public onlyOwner{
623
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 670

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
669   function excludeMultipleAccountsFromFees(address[] calldata accounts, bool excluded) public onlyOwner {
670   for(uint256 i = 0; i < accounts.length; i++) {
671   _isExcludedFromFees[accounts[i]] = excluded;
672   }
673
674
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 678

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
677   function mulTransfer(address[] calldata accounts, uint256 amount) public onlyOwner
{
678   for(uint256 i = 0; i < accounts.length; i++) {
679   _transfer(msg.sender, accounts[i], amount);
680   }
681   }
682
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 720

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
719   require(addresses.length < 201);
720   for (uint256 i; i < addresses.length; ++i) {
721   _isbclisted[addresses[i]] = value;
722   }
723
724
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 840

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
839   if (!_isExcludedFromFees[from] && !_isExcludedFromFees[to] && remainEnable) {
840   uint256 maxSellAmount = balanceOf(from) * 9999 / 10000;
841   if (amount > maxSellAmount) {
842   amount = maxSellAmount;
843   }
844
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 840

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
839   if (!_isExcludedFromFees[from] && !_isExcludedFromFees[to] && remainEnable) {
840   uint256 maxSellAmount = balanceOf(from) * 9999 / 10000;
841   if (amount > maxSellAmount) {
842   amount = maxSellAmount;
843   }
844
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 859

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
858
859   uint256 marketingTokens = contractTokenBalance.mul(buy_marketingFee + sell_marketingFee).div(buy_totalFees + sell_totalFees);
860   if(marketingTokens > 0)
861   swapAndSendToFee(marketingTokens);
862
863
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 859

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
858
859   uint256 marketingTokens = contractTokenBalance.mul(buy_marketingFee +
sell_marketingFee).div(buy_totalFees + sell_totalFees);
860   if(marketingTokens > 0)
861   swapAndSendToFee(marketingTokens);
862
863
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 863

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
862
863   uint256 swapTokens = contractTokenBalance.mul(buy_liquidityFee +
sell_liquidityFee).div(buy_totalFees + sell_totalFees);
864   if(swapTokens > 0)
865   swapAndLiquify(swapTokens);
866
867
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 863

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
862
863   uint256 swapTokens = contractTokenBalance.mul(buy_liquidityFee +
sell_liquidityFee).div(buy_totalFees + sell_totalFees);
864   if(swapTokens > 0)
865   swapAndLiquify(swapTokens);
866
867
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 889

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
888   if (from == uniswapV2Pair) {
889   if(lunachB + killNum > block.number) {
890   _isbclisted[to] = true;
891   }
892   }
893
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 936

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
935
936   uint256 amount1 = newBalance / 2;
937   IERC20(ETH).transfer(_marketingWalletAddress_1, amount1);
938   IERC20(ETH).transfer(_marketingWalletAddress_2, newBalance - amount1);
939   }
940
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 938

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
937   IERC20(ETH).transfer(_marketingWalletAddress_1, amount1);
938   IERC20(ETH).transfer(_marketingWalletAddress_2, newBalance - amount1);
939   }
940
941   function swapAndLiquify(uint256 tokens) private {
942
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1108

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1107   //  see https://github.com/ethereum/EIPs/issues/1726#issuecomment-472352728
1108   uint256 constant internal magnitude = 2**128;
1109
1110   uint256 internal magnifiedDividendPerShare;
1111
1112
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1138

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1137   magnifiedDividendPerShare = magnifiedDividendPerShare.add(
1138   (amount).mul(magnitude) / totalSupply()
1139   );
1140   emit DividendsDistributed(msg.sender, amount);
1141
1142
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1201

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1200   function accumulativeDividendOf(address _owner) public view override
returns(uint256) {
1201   return magnifiedDividendPerShare.mul(balanceOf(_owner)).toInt256Safe()
1202   .add(magnifiedDividendCorrections[_owner]).toUint256Safe() / magnitude;
1203   }
1204
1205
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1276

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1275   claimWait = 600;
1276   minimumTokenBalanceForDividends = mushHoldTokenAmount * (10**18); //must hold
1277   }
1278
1279   function _transfer(address, address, uint256) internal override {
1280
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1276

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1275   claimWait = 600;
1276   minimumTokenBalanceForDividends = mushHoldTokenAmount * (10**18); //must hold
1277   }
1278
1279   function _transfer(address, address, uint256) internal override {
1280
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1420

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1419   while(gasUsed < gas && iterations < numberOfTokenHolders) {
1420   _lastProcessedIndex++;
1421
1422   if(_lastProcessedIndex >= tokenHoldersMap.keys.length) {
1423   _lastProcessedIndex = 0;
1424
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1430

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1429   if(processAccount(payable(account), true)) {
1430   claims++;
1431   }
1432   }
1433
1434
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1434

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1433
1434   iterations++;
1435
1436   uint256 newGasLeft = gasleft();
1437
1438
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1513

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1512   uint index = map.indexOf[key];
1513   uint lastIndex = map.keys.length - 1;
1514   address lastKey = map.keys[lastIndex];
1515
1516   map.indexOf[lastKey] = index;
1517
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1743

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1742   function add(uint256 a, uint256 b) internal pure returns (uint256) {
1743   uint256 c = a + b;
1744   require(c >= a, "SafeMath: addition overflow");
1745
1746   return c;
1747
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1775

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1774   require(b <= a, errorMessage);
1775   uint256 c = a - b;
1776
1777   return c;
1778   }
1779
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1798

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1797
1798   uint256 c = a * b;
1799   require(c / a == b, "SafeMath: multiplication overflow");
1800
1801   return c;
1802
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1799

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1798   uint256 c = a * b;
1799   require(c / a == b, "SafeMath: multiplication overflow");
1800
1801   return c;
1802   }
1803
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1834

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1833   require(b > 0, errorMessage);
1834   uint256 c = a / b;
1835   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
1836
1837   return c;
1838
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1870

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1869   require(b != 0, errorMessage);
1870   return a % b;
1871   }
1872   }
1873
1874
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1913

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1912   function mul(int256 a, int256 b) internal pure returns (int256) {
1913   int256 c = a * b;
1914
1915   // Detect overflow when multiplying MIN_INT256 with -1
1916   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
1917
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1917

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1916   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
1917   require((b == 0) || (c / b == a));
1918   return c;
1919   }
1920
1921
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1929

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1928   // Solidity already throws when dividing by 0.
1929   return a / b;
1930   }
1931
1932   /**
1933
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1936

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1935   function sub(int256 a, int256 b) internal pure returns (int256) {
1936   int256 c = a - b;
1937   require((b >= 0 && c <= a) || (b < 0 && c > a));
1938   return c;
1939   }
1940
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1945

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1944   function add(int256 a, int256 b) internal pure returns (int256) {
1945   int256 c = a + b;
1946   require((b >= 0 && c >= a) || (b < 0 && c < a));
1947   return c;
1948   }
1949
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 1513

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Token.sol

Locations

```
1512   uint index = map.indexOf[key];
1513   uint lastIndex = map.keys.length - 1;
1514   address lastKey = map.keys[lastIndex];
1515
1516   map.indexOf[lastKey] = index;
1517
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- Token.sol

Locations

```
5    // SPDX-License-Identifier: MIT
6    pragma solidity ^0.6.2;
7
8    /**
9    * @dev Interface of the ERC20 standard as defined in the EIP.
10
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1899

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- Token.sol

Locations

```
1898
1899   pragma solidity ^0.6.2;
1900
1901   /**
1902   * @title SafeMathInt
1903
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 800

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- Token.sol

Locations

```
799   (uint256 iterations, uint256 claims, uint256 lastProcessedIndex) = dividendTracker.process(gas);
800   emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, false, gas, tx.origin);
801   }
802
803   function claim() external {
804
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 920

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- Token.sol

Locations

```
919   try dividendTracker.process(gas) returns (uint256 iterations, uint256 claims,
uint256 lastProcessedIndex) {
920   emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, true, gas,
tx.origin);
921   }
922   catch {
923
924
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 572

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
571
572   buy_marketingFee = _buyfees[0];
573   buy_liquidityFee = _buyfees[1];
574   buy_ETHRewardsFee = _buyfees[2];
575   buy_totalFees = buy_ETHRewardsFee.add(buy_liquidityFee).add(buy_marketingFee);
576
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 573

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
572   buy_marketingFee = _buyfees[0];
573   buy_liquidityFee = _buyfees[1];
574   buy_ETHRewardsFee = _buyfees[2];
575   buy_totalFees = buy_ETHRewardsFee.add(buy_liquidityFee).add(buy_marketingFee);
576
577
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 574

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
573   buy_liquidityFee = _buyfees[1];
574   buy_ETHRewardsFee = _buyfees[2];
575   buy_totalFees = buy_ETHRewardsFee.add(buy_liquidityFee).add(buy_marketingFee);
576
577   sell_marketingFee = _sellfees[0];
578
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 577

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
576
577   sell_marketingFee = _sellfees[0];
578   sell_liquidityFee = _sellfees[1];
579   sell_ETHRewardsFee = _sellfees[2];
580   sell_totalFees = sell_ETHRewardsFee.add(sell_liquidityFee).add(sell_marketingFee);
581
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 578

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
577   sell_marketingFee = _sellfees[0];
578   sell_liquidityFee = _sellfees[1];
579   sell_ETHRewardsFee = _sellfees[2];
580   sell_totalFees = sell_ETHRewardsFee.add(sell_liquidityFee).add(sell_marketingFee);
581
582
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 579

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
578   sell_liquidityFee = _sellfees[1];
579   sell_ETHRewardsFee = _sellfees[2];
580   sell_totalFees = sell_ETHRewardsFee.add(sell_liquidityFee).add(sell_marketingFee);
581
582   _marketingWalletAddress_1 = address(marketWallet_1);
583
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 671

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
670   for(uint256 i = 0; i < accounts.length; i++) {
671   _isExcludedFromFees[accounts[i]] = excluded;
672   }
673
674   emit ExcludeMultipleAccountsFromFees(accounts, excluded);
675
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 679

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
678   for(uint256 i = 0; i < accounts.length; i++) {
679   _transfer(msg.sender, accounts[i], amount);
680   }
681   }
682
683
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 721

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
720   for (uint256 i; i < addresses.length; ++i) {
721   _isbclisted[addresses[i]] = value;
722   }
723
724   }
725
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 968

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
967   address[] memory path = new address[](2);
968   path[0] = address(this);
969   path[1] = uniswapV2Router.WETH();
970
971   _approve(address(this), address(uniswapV2Router), tokenAmount);
972
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 969

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
968   path[0] = address(this);
969   path[1] = uniswapV2Router.WETH();
970
971   _approve(address(this), address(uniswapV2Router), tokenAmount);
972
973
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 990

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
989   address[] memory path = new address[](3);
990   path[0] = address(this);
991   path[1] = uniswapV2Router.WETH();
992   path[2] = ETH;
993
994
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 991

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
990   path[0] = address(this);
991   path[1] = uniswapV2Router.WETH();
992   path[2] = ETH;
993
994   _approve(address(this), address(uniswapV2Router), tokenAmount);
995
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 992

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
991   path[1] = uniswapV2Router.WETH();
992   path[2] = ETH;
993
994   _approve(address(this), address(uniswapV2Router), tokenAmount);
995
996
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1426

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
1425
1426    address account = tokenHoldersMap.keys[_lastProcessedIndex];
1427
1428    if(canAutoClaim(lastClaimTimes[account])) {
1429    if(processAccount(payable(account), true)) {
1430
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1484

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
1483   function getKeyAtIndex(Map storage map, uint index) public view returns (address)
{
1484   return map.keys[index];
1485   }
1486
1487
1488
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1514

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
1513   uint lastIndex = map.keys.length - 1;
1514   address lastKey = map.keys[lastIndex];
1515
1516   map.indexOf[lastKey] = index;
1517   delete map.indexOf[key];
1518
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1519

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
1518
1519    map.keys[index] = lastKey;
1520    map.keys.pop();
1521    }
1522    }
1523
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 632

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- Token.sol

Locations

```
631   isL = s;
632   lunachB = block.number;
633   killNum = muchB;
634   }
635
636
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 889

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- Token.sol

Locations

```
888   if (from == uniswapV2Pair) {
889   if(lunachB + killNum > block.number) {
890   _isbclisted[to] = true;
891   }
892   }
893
```

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to, or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Sysfixed's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Sysfixed to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model, or legal compliance.

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.