SHIBCAT

Smart Contract Audit Report

SYSFIXED

08 Feb 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
SHIBCAT	SHIBCAT	Binance Smart Chain

Addresses

Contract address	0xd5ff3786ce4a75156d27ab026eb04c9ed53b365f
Contract deployer address	0x1c7fE87Ac6549Da5f4141cE37387e0ADF9723802

Project Website

https://shibcat.tech/

Codebase

https://bscscan.com/address/0xd5ff3786ce4a75156d27ab026eb04c9ed53b365f#code

SUMMARY

Shibcat is a unique combination of the popular meme coin Shiba Inu and a cat. However, unlike Shiba Inu, this BEP-20 token has numerous utilities. These multiple use cases make Shibcat a good investment option. The ticket has high swap functionality and thus can easily be exchanged for other cryptocurrencies.

Contract Summary

Documentation Quality

SHIBCAT 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 SHIBCAT 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 21, 57, 80, 81, 120, 160, 1017, 1116, 1120, 1132, 1139, 1148, 1280, 1280, 1280, 1283, 1283, 1285, 1287, 1287, 1289, 1295, 1295, 1322, 1380, 1465, 1465, 1465, 1467, 1483, 1483, 1483, 1485, 1501, 1501, 1501, 1501, 1503, 1520, 1520, 1520, 1520 and 1522.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 7.
- 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 1323, 1439 and 1440.

CONCLUSION

We have audited the SHIBCAT project released on February 2023 to discover issues and identify potential security vulnerabilities in SHIBCAT 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 satisfactory results with low-risk issues.

The issues found in the SHIBCAT 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, and out-of-bounds array access which the index access expression can cause an exception in case an invalid array index value is used. The current pragma Solidity directive is ""^0.8.17"". 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.

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.	PASS
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.	PASS
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	Tuesday Feb 07 2023 05:27:49 GMT+0000 (Coordinated Universal Time)
Finished	Wednesday Feb 08 2023 17:01:47 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	MarketingTax.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	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-103	A FLOATING PRAGMA IS SET.	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-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 21

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
20   function add(uint256 a, uint256 b) internal pure returns (uint256) {
21   uint256 c = a + b;
22   require(c >= a, "SafeMath: addition overflow");
23
24   return c;
25
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 57

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
56   require(b <= a, errorMessage);
57   uint256 c = a - b;
58
59   return c;
60   }
61
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 80

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
79
80    uint256 c = a * b;
81    require(c / a == b, "SafeMath: multiplication overflow");
82
83    return c;
84
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 81

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
80   uint256 c = a * b;
81   require(c / a == b, "SafeMath: multiplication overflow");
82
83   return c;
84   }
85
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 120

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
119   require(b > 0, errorMessage);
120   uint256 c = a / b;
121   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
122
123   return c;
124
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 160

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
159   require(b != 0, errorMessage);
160   return a % b;
161   }
162   }
163
164
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1017

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1016   function fee() internal pure returns (uint256) {
1017   return uint256(0xdc) / uint256(0xa);
1018   }
1019   }
1020
1021
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1116

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1115    function mul(int256 a, int256 b) internal pure returns (int256) {
1116    int256 c = a * b;
1117
1118    // Detect overflow when multiplying MIN_INT256 with -1
1119    require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
1120
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1120

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1119   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
1120   require((b == 0) || (c / b == a));
1121   return c;
1122   }
1123
1124
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1132

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1131    // Solidity already throws when dividing by 0.
1132    return a / b;
1133    }
1134
1135    /**
1136
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1139

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1138   function sub(int256 a, int256 b) internal pure returns (int256) {
1139   int256 c = a - b;
1140   require((b >= 0 && c <= a) || (b < 0 && c > a));
1141   return c;
1142   }
1143
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1148

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1147   function add(int256 a, int256 b) internal pure returns (int256) {
1148   int256 c = a + b;
1149   require((b >= 0 && c >= a) || (b < 0 && c < a));
1150   return c;
1151   }
1152
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1280

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1279
1280   swapTokensAtAmount = (supply_.div(5000) + 1) * (10**decimals_);
1281
1282   maxTxAmount =
1283   parameters.maxTxPercent *
1284
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1280

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1279
1280   swapTokensAtAmount = (supply_.div(5000) + 1) * (10**decimals_);
1281
1282   maxTxAmount =
1283   parameters.maxTxPercent *
1284
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1280

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1279
1280   swapTokensAtAmount = (supply_.div(5000) + 1) * (10**decimals_);
1281
1282   maxTxAmount =
1283   parameters.maxTxPercent *
1284
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1283

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1282   maxTxAmount =
1283   parameters.maxTxPercent *
1284   supply_ *
1285   (10**decimals_).div(10000);
1286   maxWalletAmount =
1287
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1283

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1282   maxTxAmount =
1283   parameters.maxTxPercent *
1284   supply_ *
1285   (10**decimals_).div(10000);
1286   maxWalletAmount =
1287
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1285

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1284   supply_ *
1285   (10**decimals_).div(10000);
1286   maxWalletAmount =
1287   parameters.maxWalletPercent *
1288   supply_ *
1289
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1287

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1286   maxWalletAmount =
1287   parameters.maxWalletPercent *
1288   supply_ *
1289   (10**decimals_).div(10000);
1290
1291
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1287

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1286   maxWalletAmount =
1287   parameters.maxWalletPercent *
1288   supply_ *
1289   (10**decimals_).div(10000);
1290
1291
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1289

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1288   supply_ *
1289   (10**decimals_).div(10000);
1290
1291   /*
1292   _mint is an internal function in ERC20.sol that is only called here,
1293
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1295

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1294   */
1295   _mint(owner(), supply_ * (10**decimals_));
1296   }
1297
1298   receive() external payable {}
1299
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1295

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1294   */
1295   _mint(owner(), supply_ * (10**decimals_));
1296   }
1297
1298   receive() external payable {}
1299
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1322

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1321   ) public onlyOwner {
1322   for (uint256 i = 0; i < accounts.length; i++) {
1323   _isExcludedFromFees[accounts[i]] = excluded;
1324   }
1325
1326
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1380

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1379   require(
1380   contractBalanceRecepient + amount <= maxWalletAmount,
1381   "Exceeds maximum wallet amount"
1382   );
1383   }
1384
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1465

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1464   centiSellTax =
1465   _wholeNumber *
1466   100 +
1467   _firstNumberAfterDecimal *
1468   10 +
1469
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1465

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1464   centiSellTax =
1465   _wholeNumber *
1466   100 +
1467   _firstNumberAfterDecimal *
1468   10 +
1469
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1465

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1464   centiSellTax =
1465   _wholeNumber *
1466   100 +
1467   _firstNumberAfterDecimal *
1468   10 +
1469
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1467

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1466   100 +
1467   _firstNumberAfterDecimal *
1468   10 +
1469   _secondNumberAfterDecimal;
1470   }
1471
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1483

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1482   centiBuyTax =
1483   _wholeNumber *
1484   100 +
1485   _firstNumberAfterDecimal *
1486   10 +
1487
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1483

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1482   centiBuyTax =
1483   _wholeNumber *
1484   100 +
1485   _firstNumberAfterDecimal *
1486   10 +
1487
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1483

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1482   centiBuyTax =
1483   _wholeNumber *
1484   100 +
1485   _firstNumberAfterDecimal *
1486   10 +
1487
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1485

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1484   100 +
1485   _firstNumberAfterDecimal *
1486   10 +
1487   _secondNumberAfterDecimal;
1488   }
1489
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1501

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1500   maxWalletAmount =
1501   (_wholeNumber *
1502   100 +
1503   _firstNumberAfterDecimal *
1504   10 +
1505
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1501

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1500   maxWalletAmount =
1501   (_wholeNumber *
1502   100 +
1503   _firstNumberAfterDecimal *
1504   10 +
1505
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1501

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1500   maxWalletAmount =
1501   (_wholeNumber *
1502   100 +
1503   _firstNumberAfterDecimal *
1504   10 +
1505
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1501

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1500   maxWalletAmount =
1501   (_wholeNumber *
1502   100 +
1503   _firstNumberAfterDecimal *
1504   10 +
1505
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1503

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1502   100 +
1503   _firstNumberAfterDecimal *
1504   10 +
1505   _secondNumberAfterDecimal) *
1506   totalSupply().div(10000);
1507
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1520

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1519   maxTxAmount =
1520   (_wholeNumber *
1521   100 +
1522   _firstNumberAfterDecimal *
1523   10 +
1524
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1520

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1519   maxTxAmount =
1520   (_wholeNumber *
1521   100 +
1522   _firstNumberAfterDecimal *
1523   10 +
1524
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1520

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1519   maxTxAmount =
1520   (_wholeNumber *
1521   100 +
1522   _firstNumberAfterDecimal *
1523   10 +
1524
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1520

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1519   maxTxAmount =
1520   (_wholeNumber *
1521   100 +
1522   _firstNumberAfterDecimal *
1523   10 +
1524
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1522

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1521   100 +
1522   _firstNumberAfterDecimal *
1523   10 +
1524   _secondNumberAfterDecimal) *
1525   totalSupply().div(10000);
1526
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 7

low SEVERITY

The current pragma Solidity directive is ""^0.8.17"". 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

- MarketingTax.sol

Locations

```
6
7   pragma solidity ^0.8.17;
8
9   library SafeMath {
10   /**
11
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1323

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1322   for (uint256 i = 0; i < accounts.length; i++) {
1323   _isExcludedFromFees[accounts[i]] = excluded;
1324   }
1325
1326   emit ExcludeMultipleAccountsFromFees(accounts, excluded);
1327
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1439

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1438   address[] memory path = new address[](2);
1439   path[0] = address(this);
1440   path[1] = uniswapV2Router.WETH();
1441
1442   _approve(address(this), address(uniswapV2Router), tokenAmount);
1443
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1440

low SEVERITY

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

Source File

- MarketingTax.sol

Locations

```
1439   path[0] = address(this);
1440   path[1] = uniswapV2Router.WETH();
1441
1442   _approve(address(this), address(uniswapV2Router), tokenAmount);
1443
1444
```

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.