Tuzi2023

Smart Contract Audit Report

SYSFIXED

08 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Tuzi2023	T2023	Binance Smart Chain

Addresses

Contract address	0xD03cc658eF2192d7cE174b91b752E9A72821dd65
Contract deployer address	0x3519400E403a0260dA97B0a29DB5c6C22718e994

Project Website

https://www.tuzi2023.com/

Codebase

https://bscscan.com/address/0xD03cc658eF2192d7cE174b91b752E9A72821dd65#code

SUMMARY

Tuzi is a meme token made in celebration of new year of china. Ring in the year of rabbit with doge coin rewards from our Tuzi Token. Dev team always keep Community Connection. Confident to use the power of community for pushing in organic way. Long-term development. Give up ownership after launch.

Contract Summary

Documentation Quality

Tuzi2023 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 Tuzi2023 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 60, 188, 220, 243, 244, 279, 315, 331, 335, 347, 354, 363, 891, 930, 976, 1011, 1098, 1098, 1175, 1175, 1244, 1273, 1273, 1426, 1426, 1426, 1456, 1459, 1459, 1462, 1462, 1463, 1463, 1463, 1469, 1469, 1472, 1502, 1503, 1503, 1504, 1510, 1511, 1511, 1512, 1520, 1520, 1526, 1526, 1532, 1532, 1596, 1596, 1608, 1608, 1678, 1687, 1765, 1775, 1779, 1784 and 60.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 8, 74, 121, 174, 398, 422, 481, 586, 875 and 1066.
- 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 31, 61, 66, 973, 974, 1245, 1541, 1542 and 1771.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 1201 and 1494.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 1285 and 1456.

CONCLUSION

We have audited the Tuzi2023 project released on January 2023 to discover issues and identify potential security vulnerabilities in Tuzi2023 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 Tuzi2023 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, 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.

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
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
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	Delegate calls 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
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
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

SMART CONTRACT ANALYSIS

Started	Saturday Jan 07 2023 12:01:21 GMT+0000 (Coordinated Universal Time)
Finished	Sunday Jan 08 2023 19:15:18 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Tuzi2023.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-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-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged

SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	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-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 60

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
59   uint index = map.indexOf[key];
60   uint lastIndex = map.keys.length - 1;
61   address lastKey = map.keys[lastIndex];
62
63   map.indexOf[lastKey] = index;
64
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 188

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
187   function add(uint256 a, uint256 b) internal pure returns (uint256) {
188   uint256 c = a + b;
189   require(c >= a, "SafeMath: addition overflow");
190
191   return c;
192
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 220

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
219   require(b <= a, errorMessage);
220   uint256 c = a - b;
221
222   return c;
223   }
224
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 243

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
242
243   uint256 c = a * b;
244   require(c / a == b, "SafeMath: multiplication overflow");
245
246   return c;
247
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 244

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
243   uint256 c = a * b;
244   require(c / a == b, "SafeMath: multiplication overflow");
245
246   return c;
247   }
248
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 279

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
278   require(b > 0, errorMessage);
279   uint256 c = a / b;
280   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
281
282   return c;
283
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 315

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
314   require(b != 0, errorMessage);
315   return a % b;
316   }
317   }
318
319
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 331

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
330   function mul(int256 a, int256 b) internal pure returns (int256) {
331   int256 c = a * b;
332
333   // Detect overflow when multiplying MIN_INT256 with -1
334   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
335
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 335

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
334   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
335   require((b == 0) || (c / b == a));
336   return c;
337   }
338
339
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 347

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
346   // Solidity already throws when dividing by 0.
347   return a / b;
348   }
349
350   /**
351
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 354

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
353   function sub(int256 a, int256 b) internal pure returns (int256) {
354   int256 c = a - b;
355   require((b >= 0 && c <= a) || (b < 0 && c > a));
356   return c;
357   }
358
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 363

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
362   function add(int256 a, int256 b) internal pure returns (int256) {
363   int256 c = a + b;
364   require((b >= 0 && c >= a) || (b < 0 && c < a));
365   return c;
366   }
367
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 891

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
890   //  see https://github.com/ethereum/EIPs/issues/1726#issuecomment-472352728
891   uint256 constant internal magnitude = 2**128;
892
893   IRouter public router;
894   address public rewardToken;
895
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 930

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
929   magnifiedDividendPerShare = magnifiedDividendPerShare.add(
930   (msg.value).mul(magnitude) / totalSupply()
931   );
932   emit DividendsDistributed(msg.sender, msg.value);
933
934
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 976

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
975
976   try router.swapExactETHForTokens{value: amt}(0, path, user, block.timestamp + 2){
977   return true;
978   } catch {
979   return false;
980
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1011

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1010   function accumulativeDividendOf(address _owner) public view override
returns(uint256) {
1011   return magnifiedDividendPerShare.mul(balanceOf(_owner)).toInt256Safe()
1012   .add(magnifiedDividendCorrections[_owner]).toUint256Safe() / magnitude;
1013   }
1014
1015
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1098

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1097
1098   uint256 public swapTokensAtAmount = 1e9 * 10**9;
1099
1100   string private currentRewardToken;
1101
1102
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1098

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1097
1098   uint256 public swapTokensAtAmount = 1e9 * 10**9;
1099
1100   string private currentRewardToken;
1101
1102
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1175

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1174   */
1175   _tokengeneration(owner(), 1e12 * (10**9));
1176   }
1177
1178   receive() external payable {}
1179
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1175

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1174   */
1175   _tokengeneration(owner(), 1e12 * (10**9));
1176   }
1177
1178   receive() external payable {}
1179
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1244

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1243   {
1244   for (uint256 i = 0; i < accounts.length; i++) {
1245   _isExcludedFromFees[accounts[i]] = excluded;
1246   }
1247   emit ExcludeMultipleAccountsFromFees(accounts, excluded);
1248
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1273

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1272   require(amount < 1e10,"Swap Threshold should be less than 1% of total supply");
1273   swapTokensAtAmount = amount * 10**9;
1274   }
1275
1276   /// @notice Enable or disable internal swaps
1277
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1273

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1272   require(amount < 1e10,"Swap Threshold should be less than 1% of total supply");
1273   swapTokensAtAmount = amount * 10**9;
1274   }
1275
1276   /// @notice Enable or disable internal swaps
1277
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1426

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1425   bool canSwap = contractTokenBalance >= swapTokensAtAmount;
1426   uint256 swapTax = sellTaxes.rewards +
1427   sellTaxes.marketing +
1428   sellTaxes.dev +
1429   sellTaxes.liquidity ;
1430
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1426

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1425   bool canSwap = contractTokenBalance >= swapTokensAtAmount;
1426   uint256 swapTax = sellTaxes.rewards +
1427   sellTaxes.marketing +
1428   sellTaxes.dev +
1429   sellTaxes.liquidity ;
1430
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1426

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1425   bool canSwap = contractTokenBalance >= swapTokensAtAmount;
1426   uint256 swapTax = sellTaxes.rewards +
1427   sellTaxes.marketing +
1428   sellTaxes.dev +
1429   sellTaxes.liquidity ;
1430
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1456

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1455   if (takeFee) {
1456   bool beforeTradingFee = block.number <= startTradingBlock + antiBotBlocks;
1457   uint256 swapAmt;
1458   if (automatedMarketMakerPairs[to] && !beforeTradingFee) {
1459   swapAmt = (amount * swapTax) / 100;
1460
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1459

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1458   if (automatedMarketMakerPairs[to] && !beforeTradingFee) {
1459   swapAmt = (amount * swapTax) / 100;
1460   } else if (automatedMarketMakerPairs[from] && !beforeTradingFee) {
1461   swapAmt =
1462   (amount *
1463
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1459

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1458   if (automatedMarketMakerPairs[to] && !beforeTradingFee) {
1459   swapAmt = (amount * swapTax) / 100;
1460   } else if (automatedMarketMakerPairs[from] && !beforeTradingFee) {
1461   swapAmt =
1462   (amount *
1463
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1462

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1461   swapAmt =
1462   (amount *
1463   (buyTaxes.rewards +
1464   buyTaxes.marketing +
1465   buyTaxes.dev +
1466
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1462

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1461   swapAmt =
1462   (amount *
1463   (buyTaxes.rewards +
1464   buyTaxes.marketing +
1465   buyTaxes.dev +
1466
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1463

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1462   (amount *
1463   (buyTaxes.rewards +
1464   buyTaxes.marketing +
1465   buyTaxes.dev +
1466   buyTaxes.liquidity )) /
1467
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1463

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1462   (amount *
1463   (buyTaxes.rewards +
1464   buyTaxes.marketing +
1465   buyTaxes.dev +
1466   buyTaxes.liquidity )) /
1467
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1463

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1462   (amount *
1463   (buyTaxes.rewards +
1464   buyTaxes.marketing +
1465   buyTaxes.dev +
1466   buyTaxes.liquidity )) /
1467
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1469

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1468   } else if (beforeTradingFee) {
1469   swapAmt = (amount * launchtax) / 100;
1470   }
1471
1472   amount = amount - (swapAmt);
1473
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1469

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1468   } else if (beforeTradingFee) {
1469   swapAmt = (amount * launchtax) / 100;
1470   }
1471
1472   amount = amount - (swapAmt);
1473
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1472

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1471
1472   amount = amount - (swapAmt);
1473   super._transfer(from, address(this), swapAmt);
1474   }
1475   super._transfer(from, to, amount);
1476
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1502

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1501   // Split the contract balance into halves
1502   uint256 denominator = swapTax * 2;
1503   uint256 tokensToAddLiquidityWith = (tokens * sellTaxes.liquidity) / denominator;
1504   uint256 toSwap = tokens - tokensToAddLiquidityWith;
1505
1506
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1503

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1502   uint256 denominator = swapTax * 2;
1503   uint256 tokensToAddLiquidityWith = (tokens * sellTaxes.liquidity) / denominator;
1504   uint256 toSwap = tokens - tokensToAddLiquidityWith;
1505
1506   uint256 initialBalance = address(this).balance;
1507
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1503

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1502   uint256 denominator = swapTax * 2;
1503   uint256 tokensToAddLiquidityWith = (tokens * sellTaxes.liquidity) / denominator;
1504   uint256 toSwap = tokens - tokensToAddLiquidityWith;
1505
1506   uint256 initialBalance = address(this).balance;
1507
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1504

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1503   uint256 tokensToAddLiquidityWith = (tokens * sellTaxes.liquidity) / denominator;
1504   uint256 toSwap = tokens - tokensToAddLiquidityWith;
1505
1506   uint256 initialBalance = address(this).balance;
1507
1508
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1510

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1509
1510   uint256 deltaBalance = address(this).balance - initialBalance;
1511   uint256 unitBalance = deltaBalance / (denominator - sellTaxes.liquidity);
1512   uint256 bnbToAddLiquidityWith = unitBalance * sellTaxes.liquidity;
1513
1514
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1511

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1510   uint256 deltaBalance = address(this).balance - initialBalance;
1511   uint256 unitBalance = deltaBalance / (denominator - sellTaxes.liquidity);
1512   uint256 bnbToAddLiquidityWith = unitBalance * sellTaxes.liquidity;
1513
1514   if (bnbToAddLiquidityWith > 0) {
1515
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1511

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1510   uint256 deltaBalance = address(this).balance - initialBalance;
1511   uint256 unitBalance = deltaBalance / (denominator - sellTaxes.liquidity);
1512   uint256 bnbToAddLiquidityWith = unitBalance * sellTaxes.liquidity;
1513
1514   if (bnbToAddLiquidityWith > 0) {
1515
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1512

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1511   uint256 unitBalance = deltaBalance / (denominator - sellTaxes.liquidity);
1512   uint256 bnbToAddLiquidityWith = unitBalance * sellTaxes.liquidity;
1513
1514   if (bnbToAddLiquidityWith > 0) {
1515   // Add liquidity to pancake
1516
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1520

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1519   // Send BNB to marketingWallet
1520   uint256 marketingWalletAmt = unitBalance * 2 * sellTaxes.marketing;
1521   if (marketingWalletAmt > 0) {
1522   payable(marketingWallet).sendValue(marketingWalletAmt);
1523   }
1524
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1520

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1519   // Send BNB to marketingWallet
1520   uint256 marketingWalletAmt = unitBalance * 2 * sellTaxes.marketing;
1521   if (marketingWalletAmt > 0) {
1522   payable(marketingWallet).sendValue(marketingWalletAmt);
1523   }
1524
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1526

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1525   // Send BNB to devWallet
1526   uint256 devWalletAmt = unitBalance * 2 * sellTaxes.dev;
1527   if (devWalletAmt > 0) {
1528   payable(devWallet).sendValue(devWalletAmt);
1529   }
1530
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1526

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1525   // Send BNB to devWallet
1526   uint256 devWalletAmt = unitBalance * 2 * sellTaxes.dev;
1527   if (devWalletAmt > 0) {
1528   payable(devWallet).sendValue(devWalletAmt);
1529   }
1530
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1532

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1531   // Send BNB to rewardsContract
1532   uint256 dividends = unitBalance * 2 * sellTaxes.rewards;
1533   if (dividends > 0) {
1534   (bool success, ) = address(dividendTracker).call{ value: dividends }("");
1535   if (success) emit SendDividends(tokens, dividends);
1536
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1532

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1531   // Send BNB to rewardsContract
1532   uint256 dividends = unitBalance * 2 * sellTaxes.rewards;
1533   if (dividends > 0) {
1534   (bool success, ) = address(dividendTracker).call{ value: dividends }("");
1535   if (success) emit SendDividends(tokens, dividends);
1536
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1596

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1595   claimWait = 3600;
1596   minimumTokenBalanceForDividends = 100000 * (10**decimals());
1597   }
1598
1599   function _transfer(
1600
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1596

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1595   claimWait = 3600;
1596   minimumTokenBalanceForDividends = 100000 * (10**decimals());
1597   }
1598
1599   function _transfer(
1600
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1608

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1607   function setMinBalanceForDividends(uint256 amount) external onlyOwner {
1608   minimumTokenBalanceForDividends = amount * 10**decimals();
1609   }
1610
1611   function excludeFromDividends(address account, bool value) external onlyOwner {
1612
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1608

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1607   function setMinBalanceForDividends(uint256 amount) external onlyOwner {
1608   minimumTokenBalanceForDividends = amount * 10**decimals();
1609   }
1610
1611   function excludeFromDividends(address account, bool value) external onlyOwner {
1612
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1678

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1677
1678   iterationsUntilProcessed = index + (int256(processesUntilEndOfArray));
1679   }
1680   }
1681
1682
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1687

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1686
1687   nextClaimTime = lastClaimTime > 0 ? lastClaimTime + (claimWait) : 0;
1688
1689   secondsUntilAutoClaimAvailable = nextClaimTime > block.timestamp
1690   ? nextClaimTime.sub(block.timestamp)
1691
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1765

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1764   while (gasUsed < gas && iterations < numberOfTokenHolders) {
1765   _lastProcessedIndex++;
1766
1767   if (_lastProcessedIndex >= tokenHoldersMap.keys.length) {
1768   _lastProcessedIndex = 0;
1769
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1775

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1774   if (processAccount(payable(account), true)) {
1775   claims++;
1776   }
1777   }
1778
1779
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1779

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1778
1779   iterations++;
1780
1781   uint256 newGasLeft = gasleft();
1782
1783
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1784

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1783   if (gasLeft > newGasLeft) {
1784   gasUsed = gasUsed + (gasLeft.sub(newGasLeft));
1785   }
1786
1787   gasLeft = newGasLeft;
1788
```

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

LINE 60

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
59   uint index = map.indexOf[key];
60   uint lastIndex = map.keys.length - 1;
61   address lastKey = map.keys[lastIndex];
62
63   map.indexOf[lastKey] = index;
64
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 8

low SEVERITY

The current pragma Solidity directive is ""^0.8.6"". 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

- Tuzi2023.sol

Locations

```
7
8   pragma solidity ^0.8.6;
9
10   library IterableMapping {
11   // Iterable mapping from address to uint;
12
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 74

low SEVERITY

The current pragma Solidity directive is ""^0.8.6"". 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

- Tuzi2023.sol

Locations

```
73
74   pragma solidity ^0.8.6;
75
76   interface IPair {
77   function sync() external;
78
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 121

low SEVERITY

The current pragma Solidity directive is ""^0.8.6"". 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

- Tuzi2023.sol

Locations

```
120
121   pragma solidity ^0.8.6;
122
123
124   /// @title Dividend-Paying Token Interface
125
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 174

low SEVERITY

The current pragma Solidity directive is ""^0.8.6"". 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

- Tuzi2023.sol

Locations

```
173
174   pragma solidity ^0.8.6;
175
176   library SafeMath {
177   /**
178
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 398

low SEVERITY

The current pragma Solidity directive is ""^0.8.6"". 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

- Tuzi2023.sol

Locations

```
397
398   pragma solidity ^0.8.6;
399
400   /*
401   * @dev Provides information about the current execution context, including the
402
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 422

low SEVERITY

The current pragma Solidity directive is ""^0.8.6"". 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

- Tuzi2023.sol

Locations

```
421
422   pragma solidity ^0.8.6;
423
424
425
426
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 481

low SEVERITY

The current pragma Solidity directive is ""^0.8.6"". 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

- Tuzi2023.sol

Locations

```
480
481   pragma solidity ^0.8.6;
482
483   /**
484   * @dev Interface of the ERC20 standard as defined in the EIP.
485
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 586

low SEVERITY

The current pragma Solidity directive is ""^0.8.6"". 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

- Tuzi2023.sol

Locations

```
585
586   pragma solidity ^0.8.6;
587
588
589
590
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 875

low SEVERITY

The current pragma Solidity directive is ""^0.8.6"". 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

- Tuzi2023.sol

Locations

```
874
875   pragma solidity ^0.8.6;
876
877
878
879
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1066

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

- Tuzi2023.sol

Locations

```
1065
1066   pragma solidity ^0.8.17;
1067
1068
1069
1070
```

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

LINE 1201

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

- Tuzi2023.sol

Locations

```
1200   gas,
1201   tx.origin
1202   );
1203   }
1204
1205
```

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

LINE 1494

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

- Tuzi2023.sol

Locations

```
1493   gas,
1494   tx.origin
1495   );
1496   } catch {}
1497   }
1498
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 31

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
30   function getKeyAtIndex(Map storage map, uint index) public view returns (address) {
31   return map.keys[index];
32   }
33
34
35
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 61

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
60   uint lastIndex = map.keys.length - 1;
61   address lastKey = map.keys[lastIndex];
62
63   map.indexOf[lastKey] = index;
64   delete map.indexOf[key];
65
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 66

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
65
66   map.keys[index] = lastKey;
67   map.keys.pop();
68   }
69   }
70
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 973

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
972   address[] memory path = new address[](2);
973   path[0] = router.WETH();
974   path[1] = rewardToken;
975
976   try router.swapExactETHForTokens{value: amt}(0, path, user, block.timestamp + 2){
977
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 974

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
973   path[0] = router.WETH();
974   path[1] = rewardToken;
975
976   try router.swapExactETHForTokens{value: amt}(0, path, user, block.timestamp + 2){
977   return true;
978
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1245

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1244   for (uint256 i = 0; i < accounts.length; i++) {
1245   _isExcludedFromFees[accounts[i]] = excluded;
1246   }
1247   emit ExcludeMultipleAccountsFromFees(accounts, excluded);
1248   }
1249
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1541

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1540   address[] memory path = new address[](2);
1541   path[0] = address(this);
1542   path[1] = router.WETH();
1543
1544   _approve(address(this), address(router), tokenAmount);
1545
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1542

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1541   path[0] = address(this);
1542   path[1] = router.WETH();
1543
1544   _approve(address(this), address(router), tokenAmount);
1545
1546
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1771

low SEVERITY

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

Source File

- Tuzi2023.sol

Locations

```
1770
1771   address account = tokenHoldersMap.keys[_lastProcessedIndex];
1772
1773   if (canAutoClaim(lastClaimTimes[account])) {
1774   if (processAccount(payable(account), true)) {
1775
```

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

LINE 1285

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

- Tuzi2023.sol

Locations

```
1284   tradingEnabled = true;
1285   startTradingBlock = block.number;
1286   }
1287
1288   function setAntiBotBlocks(uint256 numberOfBlocks) external onlyOwner{
1289
```

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

LINE 1456

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

- Tuzi2023.sol

Locations

```
1455   if (takeFee) {
1456   bool beforeTradingFee = block.number <= startTradingBlock + antiBotBlocks;
1457   uint256 swapAmt;
1458   if (automatedMarketMakerPairs[to] && !beforeTradingFee) {
1459   swapAmt = (amount * swapTax) / 100;
1460
```

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.