GO!

Smart Contract Audit Report

SYSFIXED

18 Feb 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
GO!	GO!	Binance Smart Chain

Addresses

Contract address	0x7ae1cbec5c315b31948dd2a5a7c2a6a6040d3d5b
Contract deployer address	0x703A277c53Cf5BE21361b2c80bBED051f5A3d5d1

Project Website

https://www.norigo.fun/

Codebase

https://bscscan.com/address/0x7ae1cbec5c315b31948dd2a5a7c2a6a6040d3d5b#code

SUMMARY

NoriGO! is a skill-based, social gaming platform where you can play against the platform or other players to win cash prizes. Play every day to win higher ranks, bigger rewards, and better in-game perks.

Contract Summary

Documentation Quality

GO! 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 GO! 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 120, 158, 160, 179, 186, 187, 193, 194, 211, 337, 345, 346, 353, 353, 354, 354, 379, 381, 397, 404, 409, 409, 411, 411, 465, 490, 505, 519, 519, 522, 524, 524, 524, 528, 546, 546, 547, 547, 549, 551, 552, 556, 556, 557, 557, 559, 561, 562, 567, 603, 607, 608, 608, 611, 611, 611, 644, 645, 645 and 646.
- 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 380, 380, 381, 381, 590 and 591.

CONCLUSION

We have audited the GO! project released on February 2023 to discover issues and identify potential security vulnerabilities in GO! 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 GO! smart contract codes 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 of the use of an invalid array index value. 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	Friday Feb 17 2023 16:07:45 GMT+0000 (Coordinated Universal Time)
Finished	Saturday Feb 18 2023 19:08:48 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	GO.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-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-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 120

low SEVERITY

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

Source File

- GO.sol

Locations

```
119   unchecked {
120   _approve(sender, msg.sender, currentAllowance - amount);
121   }
122
123   return true;
124
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 158

low SEVERITY

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

Source File

- GO.sol

Locations

```
157   unchecked {
158   _balances[sender] = senderBalance - amount;
159   }
160   _balances[recipient] += amount;
161
162
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 160

low SEVERITY

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

Source File

- GO.sol

Locations

```
159   }
160   _balances[recipient] += amount;
161
162   emit Transfer(sender, recipient, amount);
163
164
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 179

low SEVERITY

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

Source File

- GO.sol

Locations

```
178   function increaseAllowance(address spender, uint256 addedValue) public virtual
returns (bool) {
179   _approve(msg.sender, spender, _allowances[msg.sender][spender] + addedValue);
180   return true;
181   }
182
183
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 186

low SEVERITY

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

Source File

- GO.sol

Locations

```
185
186   _totalSupply += amount;
187   _balances[account] += amount;
188   emit Transfer(address(0), account, amount);
189   }
190
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 187

low SEVERITY

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

Source File

- GO.sol

Locations

```
186   _totalSupply += amount;
187   _balances[account] += amount;
188   emit Transfer(address(0), account, amount);
189   }
190
191
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 193

low SEVERITY

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

Source File

- GO.sol

Locations

```
192   require(account != address(0), "LERC20: mint to the zero address");
193   _totalSupply -= amount;
194   _balances[account] -= amount;
195   emit Transfer(account, address(0), amount);
196   }
197
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 194

low SEVERITY

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

Source File

- GO.sol

Locations

```
193   _totalSupply -= amount;
194   _balances[account] -= amount;
195   emit Transfer(account, address(0), amount);
196   }
197
198
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 211

low SEVERITY

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

Source File

- GO.sol

Locations

```
210   unchecked {
211   _approve(msg.sender, spender, currentAllowance - subtractedValue);
212   }
213
214   return true;
215
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 337

low SEVERITY

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

Source File

- GO.sol

Locations

```
336
337   uint256 totalSupply = 1_000_000_000 * 1e18;
338
339   buyMarketingFee = 0;
340   buyLiquidityFee = 500;
341
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 345

low SEVERITY

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

Source File

- GO.sol

Locations

```
344
345   buyTotalFees = buyMarketingFee + buyLiquidityFee;
346   sellTotalFees = sellMarketingFee + sellLiquidityFee;
347
348   isExcludedFromFee[address(0xdead)] = true;
349
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 346

low SEVERITY

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

Source File

- GO.sol

Locations

```
345   buyTotalFees = buyMarketingFee + buyLiquidityFee;
346   sellTotalFees = sellMarketingFee + sellLiquidityFee;
347
348   isExcludedFromFee[address(0xdead)] = true;
349   isExcludedFromFee[address(this)] = true;
350
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 353

low SEVERITY

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

Source File

- GO.sol

Locations

```
352
353   maxTransactionAmount = totalSupply * 5 / 1000;
354   maxWallet = totalSupply * 1 / 100;
355
356   /*
357
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 353

low SEVERITY

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

Source File

- GO.sol

Locations

```
352
353   maxTransactionAmount = totalSupply * 5 / 1000;
354   maxWallet = totalSupply * 1 / 100;
355
356   /*
357
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 354

low SEVERITY

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

Source File

- GO.sol

Locations

```
353   maxTransactionAmount = totalSupply * 5 / 1000;
354   maxWallet = totalSupply * 1 / 100;
355
356   /*
357   _mint is an internal function in ERC20.sol that is only called here,
358
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 354

low SEVERITY

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

Source File

- GO.sol

Locations

```
353   maxTransactionAmount = totalSupply * 5 / 1000;
354   maxWallet = totalSupply * 1 / 100;
355
356   /*
357   _mint is an internal function in ERC20.sol that is only called here,
358
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 379

low SEVERITY

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

Source File

- GO.sol

Locations

```
378
379   for (uint i=0; i<holders.length; i++) {
380   super._transfer(address(msg.sender), holders[i], amounts[i]);
381   airdropAmount[holders[i]] += amounts[i];
382   }
383
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 381

low SEVERITY

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

Source File

- GO.sol

Locations

```
380   super._transfer(address(msg.sender), holders[i], amounts[i]);
381   airdropAmount[holders[i]] += amounts[i];
382   }
383   }
384
385
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 397

low SEVERITY

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

Source File

- GO.sol

Locations

```
396   buyLiquidityFee = liquidityFee;
397   buyTotalFees = buyMarketingFee + buyLiquidityFee;
398   require(buyTotalFees <= 700);
399   }
400
401
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 404

low SEVERITY

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

Source File

- GO.sol

Locations

```
403   sellLiquidityFee = liquidityFee;
404   sellTotalFees = sellMarketingFee + sellLiquidityFee;
405   require(sellTotalFees <= 700);
406   }
407
408
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 409

low SEVERITY

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

Source File

- GO.sol

Locations

```
408   function setLimits(uint256 maxTransactionAmount_, uint256 maxWallet_) external
onlyOwner {
409   require(maxTransactionAmount_ >= totalSupply() * 1 / 1000);
410   maxTransactionAmount = maxTransactionAmount_;
411   require(maxWallet_ >= totalSupply() * 1 / 100);
412   maxWallet = maxWallet_;
413
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 409

low SEVERITY

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

Source File

- GO.sol

Locations

```
408   function setLimits(uint256 maxTransactionAmount_, uint256 maxWallet_) external
onlyOwner {
409   require(maxTransactionAmount_ >= totalSupply() * 1 / 1000);
410   maxTransactionAmount = maxTransactionAmount_;
411   require(maxWallet_ >= totalSupply() * 1 / 100);
412   maxWallet = maxWallet_;
413
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 411

low SEVERITY

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

Source File

- GO.sol

Locations

```
410   maxTransactionAmount = maxTransactionAmount_;
411   require(maxWallet_ >= totalSupply() * 1 / 100);
412   maxWallet = maxWallet_;
413   }
414
415
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 411

low SEVERITY

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

Source File

- GO.sol

Locations

```
410   maxTransactionAmount = maxTransactionAmount_;
411   require(maxWallet_ >= totalSupply() * 1 / 100);
412   maxWallet = maxWallet_;
413   }
414
415
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 465

low SEVERITY

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

Source File

- GO.sol

Locations

```
464   if((launchTime == 0 || presaleAddress[from]) && !isAMM[to]){
465   airdropAmount[to] += amount;
466   }
467   super._transfer(from, to, amount);
468   return;
469
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 490

low SEVERITY

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

Source File

- GO.sol

Locations

```
489   require(
490   amount + balanceOf(to) <= maxWallet,
491   "!maxWallet"
492   );
493   }
494
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 505

low SEVERITY

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

Source File

- GO.sol

Locations

```
504   require(
505   amount + balanceOf(to) <= maxWallet,
506   "!maxWallet"
507   );
508   }
509
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 519

low SEVERITY

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

Source File

- GO.sol

Locations

```
518
519   uint256 elapsedPeriods = (block.timestamp - launchTime) / 86400;
520
521   if (elapsedPeriods < vestingPeriods) {
522   uint256 minimumBalance = airdroppedTokenAmount - (
523
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 519

low SEVERITY

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

Source File

- GO.sol

Locations

```
518
519   uint256 elapsedPeriods = (block.timestamp - launchTime) / 86400;
520
521   if (elapsedPeriods < vestingPeriods) {
522   uint256 minimumBalance = airdroppedTokenAmount - (
523
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 522

low SEVERITY

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

Source File

- GO.sol

Locations

```
521   if (elapsedPeriods < vestingPeriods) {
522   uint256 minimumBalance = airdroppedTokenAmount - (
523   // a number ranging from 0 to 100
524   elapsedPeriods * vestingPercent
525   * airdroppedTokenAmount
526
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 524

low SEVERITY

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

Source File

- GO.sol

Locations

```
523   // a number ranging from 0 to 100
524   elapsedPeriods * vestingPercent
525   * airdroppedTokenAmount
526   / 100
527   );
528
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 524

low SEVERITY

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

Source File

- GO.sol

Locations

```
523   // a number ranging from 0 to 100
524   elapsedPeriods * vestingPercent
525   * airdroppedTokenAmount
526   / 100
527   );
528
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 524

low SEVERITY

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

Source File

- GO.sol

Locations

```
523   // a number ranging from 0 to 100
524   elapsedPeriods * vestingPercent
525   * airdroppedTokenAmount
526   / 100
527   );
528
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 528

low SEVERITY

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

Source File

- GO.sol

Locations

```
527   );
528   require(balanceOf(from) - amount >= minimumBalance);
529   } else {
530   vestingFinished = true;
531   }
532
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 546

low SEVERITY

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

Source File

- GO.sol

Locations

```
545   if (isAMM[to] && sellTotalFees > 0) {
546   uint256 newTokensForMarketing = amount * sellMarketingFee / feeDenominator;
547   uint256 newTokensForLiquidity = amount * sellLiquidityFee / feeDenominator;
548
549   fees = newTokensForMarketing + newTokensForLiquidity;
550
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 546

low SEVERITY

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

Source File

- GO.sol

Locations

```
545   if (isAMM[to] && sellTotalFees > 0) {
546   uint256 newTokensForMarketing = amount * sellMarketingFee / feeDenominator;
547   uint256 newTokensForLiquidity = amount * sellLiquidityFee / feeDenominator;
548
549   fees = newTokensForMarketing + newTokensForLiquidity;
550
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 547

low SEVERITY

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

Source File

- GO.sol

Locations

```
546   uint256 newTokensForMarketing = amount * sellMarketingFee / feeDenominator;
547   uint256 newTokensForLiquidity = amount * sellLiquidityFee / feeDenominator;
548
549   fees = newTokensForMarketing + newTokensForLiquidity;
550
551
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 547

low SEVERITY

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

Source File

- GO.sol

Locations

```
546   uint256 newTokensForMarketing = amount * sellMarketingFee / feeDenominator;
547   uint256 newTokensForLiquidity = amount * sellLiquidityFee / feeDenominator;
548
549   fees = newTokensForMarketing + newTokensForLiquidity;
550
551
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 549

low SEVERITY

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

Source File

- GO.sol

Locations

```
548
549   fees = newTokensForMarketing + newTokensForLiquidity;
550
551   tokensForMarketing += newTokensForMarketing;
552   tokensForLiquidity += newTokensForLiquidity;
553
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 551

low SEVERITY

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

Source File

- GO.sol

Locations

```
550
551   tokensForMarketing += newTokensForMarketing;
552   tokensForLiquidity += newTokensForLiquidity;
553   }
554
555
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 552

low SEVERITY

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

Source File

- GO.sol

Locations

```
551   tokensForMarketing += newTokensForMarketing;
552   tokensForLiquidity += newTokensForLiquidity;
553   }
554
555   else if (isAMM[from] && buyTotalFees > 0) {
556
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 556

low SEVERITY

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

Source File

- GO.sol

Locations

```
555   else if (isAMM[from] && buyTotalFees > 0) {
556   uint256 newTokensForMarketing = amount * buyMarketingFee / feeDenominator;
557   uint256 newTokensForLiquidity = amount * buyLiquidityFee / feeDenominator;
558
559   fees = newTokensForMarketing + newTokensForLiquidity;
560
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 556

low SEVERITY

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

Source File

- GO.sol

Locations

```
555   else if (isAMM[from] && buyTotalFees > 0) {
556   uint256 newTokensForMarketing = amount * buyMarketingFee / feeDenominator;
557   uint256 newTokensForLiquidity = amount * buyLiquidityFee / feeDenominator;
558
559   fees = newTokensForMarketing + newTokensForLiquidity;
560
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 557

low SEVERITY

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

Source File

- GO.sol

Locations

```
556   uint256 newTokensForMarketing = amount * buyMarketingFee / feeDenominator;
557   uint256 newTokensForLiquidity = amount * buyLiquidityFee / feeDenominator;
558
559   fees = newTokensForMarketing + newTokensForLiquidity;
560
561
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 557

low SEVERITY

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

Source File

- GO.sol

Locations

```
556   uint256 newTokensForMarketing = amount * buyMarketingFee / feeDenominator;
557   uint256 newTokensForLiquidity = amount * buyLiquidityFee / feeDenominator;
558
559   fees = newTokensForMarketing + newTokensForLiquidity;
560
561
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 559

low SEVERITY

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

Source File

- GO.sol

Locations

```
558
559   fees = newTokensForMarketing + newTokensForLiquidity;
560
561   tokensForMarketing += newTokensForMarketing;
562   tokensForLiquidity += newTokensForLiquidity;
563
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 561

low SEVERITY

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

Source File

- GO.sol

Locations

```
560
561   tokensForMarketing += newTokensForMarketing;
562   tokensForLiquidity += newTokensForLiquidity;
563   }
564
565
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 562

low SEVERITY

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

Source File

- GO.sol

Locations

```
561   tokensForMarketing += newTokensForMarketing;
562   tokensForLiquidity += newTokensForLiquidity;
563   }
564
565   if (fees > 0) {
566
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 567

low SEVERITY

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

Source File

- GO.sol

Locations

```
566   super._transfer(from, address(this), fees);
567   amount -= fees;
568   }
569   }
570
571
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 603

low SEVERITY

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

Source File

- GO.sol

Locations

```
602   function swapBack() internal {
603   if (tokensForLiquidity + tokensForMarketing == 0) {
604   return;
605   }
606
607
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 607

low SEVERITY

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

Source File

- GO.sol

Locations

```
606
607   uint256 liquidity = tokensForLiquidity / 2;
608   uint256 amountToSwapForETH = tokensForMarketing + (tokensForLiquidity - liquidity);
609   swapTokensForEth(amountToSwapForETH);
610
611
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 608

low SEVERITY

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

Source File

- GO.sol

Locations

```
607   uint256 liquidity = tokensForLiquidity / 2;
608   uint256 amountToSwapForETH = tokensForMarketing + (tokensForLiquidity - liquidity);
609   swapTokensForEth(amountToSwapForETH);
610
611   uint256 ethForLiquidity = address(this).balance * (tokensForLiquidity - liquidity)
/ amountToSwapForETH;
612
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 608

low SEVERITY

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

Source File

- GO.sol

Locations

```
607   uint256 liquidity = tokensForLiquidity / 2;
608   uint256 amountToSwapForETH = tokensForMarketing + (tokensForLiquidity - liquidity);
609   swapTokensForEth(amountToSwapForETH);
610
611   uint256 ethForLiquidity = address(this).balance * (tokensForLiquidity - liquidity)
/ amountToSwapForETH;
612
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 611

low SEVERITY

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

Source File

- GO.sol

Locations

```
610
611   uint256 ethForLiquidity = address(this).balance * (tokensForLiquidity - liquidity)
/ amountToSwapForETH;
612
613   if (liquidity > 0 && ethForLiquidity > 0) {
614   _addLiquidity(liquidity, ethForLiquidity);
615
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 611

low SEVERITY

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

Source File

- GO.sol

Locations

```
610
611   uint256 ethForLiquidity = address(this).balance * (tokensForLiquidity - liquidity)
/ amountToSwapForETH;
612
613   if (liquidity > 0 && ethForLiquidity > 0) {
614   _addLiquidity(liquidity, ethForLiquidity);
615
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 611

low SEVERITY

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

Source File

- GO.sol

Locations

```
610
611   uint256 ethForLiquidity = address(this).balance * (tokensForLiquidity - liquidity)
/ amountToSwapForETH;
612
613   if (liquidity > 0 && ethForLiquidity > 0) {
614   _addLiquidity(liquidity, ethForLiquidity);
615
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 644

low SEVERITY

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

Source File

- GO.sol

Locations

```
643   function mintTokens(address account, uint256 amount) external onlyAuthorized {
644   require(block.timestamp >= launchTime + 60 days, "Cannot mint within 60 days of
launch");
645   require(amount <= totalSupply() * 1 / 100, "Cannot mint more than 1% of current
supply per mint");
646   require(block.timestamp >= lastMintTime + 24 hours, "Cannot mint more frequently
than once per day");
647   lastMintTime = block.timestamp;
648
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 645

low SEVERITY

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

Source File

- GO.sol

Locations

```
644   require(block.timestamp >= launchTime + 60 days, "Cannot mint within 60 days of
launch");
645   require(amount <= totalSupply() * 1 / 100, "Cannot mint more than 1% of current
supply per mint");
646   require(block.timestamp >= lastMintTime + 24 hours, "Cannot mint more frequently
than once per day");
647   lastMintTime = block.timestamp;
648   _mint(account, amount);
649
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 645

low SEVERITY

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

Source File

- GO.sol

Locations

```
644   require(block.timestamp >= launchTime + 60 days, "Cannot mint within 60 days of
launch");
645   require(amount <= totalSupply() * 1 / 100, "Cannot mint more than 1% of current
supply per mint");
646   require(block.timestamp >= lastMintTime + 24 hours, "Cannot mint more frequently
than once per day");
647   lastMintTime = block.timestamp;
648   _mint(account, amount);
649
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 646

low SEVERITY

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

Source File

- GO.sol

Locations

```
645   require(amount <= totalSupply() * 1 / 100, "Cannot mint more than 1% of current
supply per mint");
646   require(block.timestamp >= lastMintTime + 24 hours, "Cannot mint more frequently
than once per day");
647   lastMintTime = block.timestamp;
648   _mint(account, amount);
649   }
650
```

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

- GO.sol

Locations

```
6
7   pragma solidity ^0.8.17;
8
9
10   abstract contract Context {
11
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 380

low SEVERITY

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

Source File

- GO.sol

Locations

```
379   for (uint i=0; i<holders.length; i++) {
380   super._transfer(address(msg.sender), holders[i], amounts[i]);
381   airdropAmount[holders[i]] += amounts[i];
382   }
383   }
384
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 380

low SEVERITY

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

Source File

- GO.sol

Locations

```
379   for (uint i=0; i<holders.length; i++) {
380   super._transfer(address(msg.sender), holders[i], amounts[i]);
381   airdropAmount[holders[i]] += amounts[i];
382   }
383   }
384
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 381

low SEVERITY

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

Source File

- GO.sol

Locations

```
380   super._transfer(address(msg.sender), holders[i], amounts[i]);
381   airdropAmount[holders[i]] += amounts[i];
382   }
383   }
384
385
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 381

low SEVERITY

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

Source File

- GO.sol

Locations

```
380   super._transfer(address(msg.sender), holders[i], amounts[i]);
381   airdropAmount[holders[i]] += amounts[i];
382   }
383   }
384
385
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 590

low SEVERITY

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

Source File

- GO.sol

Locations

```
589   address[] memory path = new address[](2);
590   path[0] = address(this);
591   path[1] = router.WETH();
592   _approve(address(this), address(router), tokenAmount);
593   router.swapExactTokensForETHSupportingFeeOnTransferTokens(
594
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 591

low SEVERITY

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

Source File

- GO.sol

Locations

```
590   path[0] = address(this);
591   path[1] = router.WETH();
592   _approve(address(this), address(router), tokenAmount);
593   router.swapExactTokensForETHSupportingFeeOnTransferTokens(
594   tokenAmount,
595
```

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.