RABBOGE INU

Smart Contract Audit Report

SYSFIXED

07 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
RABBOGE INU	RABBOGE	Binance Smart Chain

Addresses

Contract address	0xD2a610E3b73e74E263ABaa3700D1773280c60654
Contract deployer address	0xF4a7833F52B85e79451e4cd824b3Ab17ac709Ca7

Project Website

https://rabboge.com/

Codebase

https://bscscan.com/address/0xD2a610E3b73e74E263ABaa3700D1773280c60654#code

SUMMARY

Rabboge is The best of 2023 Meme (RABBIT + DOGE) Meme token. A decentralized platform that rewards $DOGE for holders. Our advantages SAFU+KYC+Audit, Dexview Trending, No Private Sale, APP Live on Google Play, Massive marketing, CMC - CG Fast Track, Tier #1 Influencers, and Promo Marketing is supported by the incubator.

Contract Summary

Documentation Quality

RABBOGE INU 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 RABBOGE INU with the discovery of several low issues.

Test Coverage

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

Audit Findings Summary

- SWC-100 SWC-108 | Explicitly define visibility for all state variables on lines 118, 119, 121, 189, 190, 192 and 203.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 133, 133, 360, 390, 463, 473, 473, 474, 479, 479, 483, 483, 484, 484, 486, 486, 487, 488, 504, 504, 582, 582, 618, 618, 619, 619, 620, 620, 663, 663, 664, 664, 679, 684, 730, 730, 732, 736, 741, 742, 742, 743 and 743.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 6.
- 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 603, 604, 742, 743 and 743.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 539.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 658.

CONCLUSION

We have audited the RABBOGE INU project released on January 2023 to discover issues and identify potential security vulnerabilities in RABBOGE INU 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 RABBOGE INU smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, a state variable visibility is not set, weak sources of randomness, tx.origin as a part of authorization control, and out of bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value. We recommend to avoid "tx.origin" issue The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead. We recommend 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.

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

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

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

SMART CONTRACT ANALYSIS

Started	Friday Jan 06 2023 17:59:28 GMT+0000 (Coordinated Universal Time)
Finished	Saturday Jan 07 2023 23:54:43 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	RABBOGEINU.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-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	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-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 133

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
132   uint8 constant private _decimals = 5;
133   uint256 constant private _tTotal = startingSupply * (10 ** _decimals);
134
135   struct Fees {
136   uint16 buyFee;
137
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 133

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
132   uint8 constant private _decimals = 5;
133   uint256 constant private _tTotal = startingSupply * (10 ** _decimals);
134
135   struct Fees {
136   uint16 buyFee;
137
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 360

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
359   if (_allowances[sender][msg.sender] != type(uint256).max) {
360   _allowances[sender][msg.sender] -= amount;
361   }
362
363   return _transfer(sender, recipient, amount);
364
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 390

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
389   if (timeSinceLastPair != 0) {
390   require(block.timestamp - timeSinceLastPair > 3 days, "3 Day cooldown.");
391   }
392   require(!lpPairs[pair], "Pair already added to list.");
393   lpPairs[pair] = true;
394
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 463

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
462   "Cannot exceed maximums.");
463   require(buyFee + sellFee <= maxRoundtripTax, "Cannot exceed roundtrip maximum.");
464   _taxRates.buyFee = buyFee;
465   _taxRates.sellFee = sellFee;
466   _taxRates.transferFee = transferFee;
467
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 473

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
472   _ratios.marketing = marketing;
473   _ratios.total = rewards + staking + marketing;
474   uint256 total = _taxRates.buyFee + _taxRates.sellFee;
475   require(_ratios.total <= total, "Cannot exceed sum of buy and sell fees.");
476   }
477
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 473

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
472   _ratios.marketing = marketing;
473   _ratios.total = rewards + staking + marketing;
474   uint256 total = _taxRates.buyFee + _taxRates.sellFee;
475   require(_ratios.total <= total, "Cannot exceed sum of buy and sell fees.");
476   }
477
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 474

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
473   _ratios.total = rewards + staking + marketing;
474   uint256 total = _taxRates.buyFee + _taxRates.sellFee;
475   require(_ratios.total <= total, "Cannot exceed sum of buy and sell fees.");
476   }
477
478
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 479

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
478   function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
returns (uint256) {
479   return((balanceOf(lpPair) * priceImpactInHundreds) / masterTaxDivisor);
480   }
481
482   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
483
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 479

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
478   function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
returns (uint256) {
479   return((balanceOf(lpPair) * priceImpactInHundreds) / masterTaxDivisor);
480   }
481
482   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
483
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 483

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
482   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
483   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
484   swapAmount = (_tTotal * amountPercent) / amountDivisor;
485   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
486   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
487
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 483

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
482   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
483   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
484   swapAmount = (_tTotal * amountPercent) / amountDivisor;
485   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
486   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
487
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 484

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
483   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
484   swapAmount = (_tTotal * amountPercent) / amountDivisor;
485   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
486   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
487   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
488
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 484

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
483   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
484   swapAmount = (_tTotal * amountPercent) / amountDivisor;
485   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
486   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
487   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
488
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 486

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
485   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
486   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
487   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
488   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
489   }
490
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 486

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
485   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
486   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
487   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
488   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
489   }
490
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 487

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
486   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
487   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
488   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
489   }
490
491
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 488

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
487   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
488   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
489   }
490
491   function setPriceImpactSwapAmount(uint256 priceImpactSwapPercent) external
onlyOwner {
492
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 504

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
503   function setRewardsProperties(uint256 _minPeriod, uint256 _minReflection, uint256
minReflectionMultiplier) external onlyOwner {
504   _minReflection = _minReflection * 10**minReflectionMultiplier;
505   cashier.setRewardsProperties(_minPeriod, _minReflection);
506   }
507
508
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 504

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
503   function setRewardsProperties(uint256 _minPeriod, uint256 _minReflection, uint256
minReflectionMultiplier) external onlyOwner {
504   _minReflection = _minReflection * 10**minReflectionMultiplier;
505   cashier.setRewardsProperties(_minPeriod, _minReflection);
506   }
507
508
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 582

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
581   uint256 swapAmt = swapAmount;
582   if (piContractSwapsEnabled) { swapAmt = (balanceOf(lpPair) * piSwapPercent) /
masterTaxDivisor; }
583   if (contractTokenBalance >= swapAmt) { contractTokenBalance = swapAmt; }
584   contractSwap(contractTokenBalance);
585   }
586
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 582

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
581   uint256 swapAmt = swapAmount;
582   if (piContractSwapsEnabled) { swapAmt = (balanceOf(lpPair) * piSwapPercent) /
masterTaxDivisor; }
583   if (contractTokenBalance >= swapAmt) { contractTokenBalance = swapAmt; }
584   contractSwap(contractTokenBalance);
585   }
586
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 618

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
617   bool success;
618   uint256 rewardsBalance = (amtBalance * ratios.rewards) / ratios.total;
619   uint256 stakingBalance = (amtBalance * ratios.staking) / ratios.total;
620   uint256 marketingBalance = amtBalance - (rewardsBalance + stakingBalance);
621
622
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 618

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
617   bool success;
618   uint256 rewardsBalance = (amtBalance * ratios.rewards) / ratios.total;
619   uint256 stakingBalance = (amtBalance * ratios.staking) / ratios.total;
620   uint256 marketingBalance = amtBalance - (rewardsBalance + stakingBalance);
621
622
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 619

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
618   uint256 rewardsBalance = (amtBalance * ratios.rewards) / ratios.total;
619   uint256 stakingBalance = (amtBalance * ratios.staking) / ratios.total;
620   uint256 marketingBalance = amtBalance - (rewardsBalance + stakingBalance);
621
622   if (ratios.rewards > 0) {
623
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 619

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
618   uint256 rewardsBalance = (amtBalance * ratios.rewards) / ratios.total;
619   uint256 stakingBalance = (amtBalance * ratios.staking) / ratios.total;
620   uint256 marketingBalance = amtBalance - (rewardsBalance + stakingBalance);
621
622   if (ratios.rewards > 0) {
623
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 620

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
619   uint256 stakingBalance = (amtBalance * ratios.staking) / ratios.total;
620   uint256 marketingBalance = amtBalance - (rewardsBalance + stakingBalance);
621
622   if (ratios.rewards > 0) {
623   try cashier.load{value: rewardsBalance}() {} catch {}
624
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 620

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
619   uint256 stakingBalance = (amtBalance * ratios.staking) / ratios.total;
620   uint256 marketingBalance = amtBalance - (rewardsBalance + stakingBalance);
621
622   if (ratios.rewards > 0) {
623   try cashier.load{value: rewardsBalance}() {} catch {}
624
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 663

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
662   allowedPresaleExclusion = false;
663   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
664   swapAmount = (balanceOf(lpPair) * 30) / 10000;
665   }
666
667
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 663

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
662   allowedPresaleExclusion = false;
663   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
664   swapAmount = (balanceOf(lpPair) * 30) / 10000;
665   }
666
667
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 664

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
663   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
664   swapAmount = (balanceOf(lpPair) * 30) / 10000;
665   }
666
667   function finalizeTransfer(address from, address to, uint256 amount, bool buy, bool
sell, bool other) internal returns (bool) {
668
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 664

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
663   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
664   swapAmount = (balanceOf(lpPair) * 30) / 10000;
665   }
666
667   function finalizeTransfer(address from, address to, uint256 amount, bool buy, bool
sell, bool other) internal returns (bool) {
668
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 679

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
678
679   _tOwned[from] -= amount;
680   uint256 amountReceived = amount;
681   if (takeFee) {
682   amountReceived = takeTaxes(from, amount, buy, sell, other);
683
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 684

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
683   }
684   _tOwned[to] += amountReceived;
685   emit Transfer(from, to, amountReceived);
686   if (!_hasLiqBeenAdded) {
687   _checkLiquidityAdd(from, to);
688
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 730

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
729   || block.chainid == 56)) { currentFee = 4500; }
730   uint256 feeAmount = amount * currentFee / masterTaxDivisor;
731   if (feeAmount > 0) {
732   _tOwned[address(this)] += feeAmount;
733   emit Transfer(from, address(this), feeAmount);
734
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 730

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
729   || block.chainid == 56)) { currentFee = 4500; }
730   uint256 feeAmount = amount * currentFee / masterTaxDivisor;
731   if (feeAmount > 0) {
732   _tOwned[address(this)] += feeAmount;
733   emit Transfer(from, address(this), feeAmount);
734
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 732

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
731   if (feeAmount > 0) {
732   _tOwned[address(this)] += feeAmount;
733   emit Transfer(from, address(this), feeAmount);
734   }
735
736
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 736

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
735
736   return amount - feeAmount;
737   }
738
739   function multiSendTokens(address[] memory accounts, uint256[] memory amounts)
external onlyOwner {
740
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 741

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
740   require(accounts.length == amounts.length, "Lengths do not match.");
741   for (uint16 i = 0; i < accounts.length; i++) {
742   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
743   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
744   }
745
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 742

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
741   for (uint16 i = 0; i < accounts.length; i++) {
742   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
743   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
744   }
745   }
746
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 742

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
741   for (uint16 i = 0; i < accounts.length; i++) {
742   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
743   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
744   }
745   }
746
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 743

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
742   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
743   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
744   }
745   }
746
747
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 743

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
742   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
743   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
744   }
745   }
746
747
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<0.9.0"". 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

- RABBOGEINU.sol

Locations

```
5   // SPDX-License-Identifier: MIT
6   pragma solidity >=0.6.0 <0.9.0;
7
8   interface IERC20 {
9   function totalSupply() external view returns (uint256);
10
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 118

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_tOwned" is internal. Other possible visibility settings are public and private.

Source File

- RABBOGEINU.sol

Locations

```
117   contract RABBOGEINU is IERC20 {
118   mapping (address => uint256) _tOwned;
119   mapping (address => bool) lpPairs;
120   uint256 private timeSinceLastPair = 0;
121   mapping (address => mapping (address => uint256)) _allowances;
122
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 119

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "lpPairs" is internal. Other possible visibility settings are public and private.

Source File

- RABBOGEINU.sol

Locations

```
118   mapping (address => uint256) _tOwned;
119   mapping (address => bool) lpPairs;
120   uint256 private timeSinceLastPair = 0;
121   mapping (address => mapping (address => uint256)) _allowances;
122   mapping (address => bool) private _isExcludedFromProtection;
123
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 121

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_allowances" is internal. Other possible visibility settings are public and private.

Source File

- RABBOGEINU.sol

Locations

```
120   uint256 private timeSinceLastPair = 0;
121   mapping (address => mapping (address => uint256)) _allowances;
122   mapping (address => bool) private _isExcludedFromProtection;
123   mapping (address => bool) private _isExcludedFromFees;
124   mapping (address => bool) private _isExcludedFromDividends;
125
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 189

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "cashier" is internal. Other possible visibility settings are public and private.

Source File

- RABBOGEINU.sol

Locations

```
188
189   Cashier cashier;
190   uint256 cashierGas = 300000;
191
192   bool inSwap;
193
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 190

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "cashierGas" is internal. Other possible visibility settings are public and private.

Source File

- RABBOGEINU.sol

Locations

```
189   Cashier cashier;
190   uint256 cashierGas = 300000;
191
192   bool inSwap;
193   bool public contractSwapEnabled = false;
194
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 192

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "inSwap" is internal. Other possible visibility settings are public and private.

Source File

- RABBOGEINU.sol

Locations

```
191
192   bool inSwap;
193   bool public contractSwapEnabled = false;
194   uint256 public swapThreshold;
195   uint256 public swapAmount;
196
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 203

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "protections" is internal. Other possible visibility settings are public and private.

Source File

- RABBOGEINU.sol

Locations

```
202   bool public _hasLiqBeenAdded = false;
203   Protections protections;
204
205   modifier inSwapFlag() {
206   inSwap = true;
207
```

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

LINE 539

low SEVERITY

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

Source File

- RABBOGEINU.sol

Locations

```
538   && to != _owner
539   && tx.origin != _owner
540   && !_liquidityHolders[to]
541   && !_liquidityHolders[from]
542   && to != DEAD
543
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 603

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
602   address[] memory path = new address[](2);
603   path[0] = address(this);
604   path[1] = dexRouter.WETH();
605
606   try dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
607
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 604

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
603   path[0] = address(this);
604   path[1] = dexRouter.WETH();
605
606   try dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
607   contractTokenBalance,
608
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 742

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
741   for (uint16 i = 0; i < accounts.length; i++) {
742   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
743   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
744   }
745   }
746
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 743

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
742   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
743   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
744   }
745   }
746
747
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 743

low SEVERITY

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

Source File

- RABBOGEINU.sol

Locations

```
742   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
743   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
744   }
745   }
746
747
```

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

LINE 658

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

- RABBOGEINU.sol

Locations

```
657   }
658   try protections.setLaunch(lpPair, uint32(block.number), uint64(block.timestamp),
_decimals) {} catch {}
659   try cashier.initialize() {} catch {}
660   tradingEnabled = true;
661   processReflect = true;
662
```

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.