



RAPPER TOKEN

Smart Contract Audit Report

TABLE OF CONTENTS

Audited Details

- Audited Project
- Blockchain
- Addresses
- Project Website
- Codebase

Summary

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

Conclusion

Audit Results

Smart Contract Analysis

- Detected Vulnerabilities

Disclaimer

About Us

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
RAPPER TOKEN	RAPT	BSC

Addresses

Contract address	0xB4B1d689077BF8b897D6B706d6aAC7597675A740
Contract deployer address	0x45E543ec2c3500c954EDb4134fC4f6871379767b

Project Website

<https://www.rappertoken.com/>

Codebase

<https://bscscan.com/address/0xB4B1d689077BF8b897D6B706d6aAC7597675A740#code>

SUMMARY

\$RAPT rewarding its investors with passive income with 2 automatic earnings. Listen to Rap And Earn Money For Free!The Concept Of L2E Is Based On Making Ordinary Things Profitable. By introducing Rapper Token, Folks Are Able To Get Paid For Listening To Rap. Listen To Earn Platform Live, KYC+AUDI, BUSD Rewards, No Private Sale, Staking,YouTube/Twitter Marketing Campaign, CEX Listing, Chinese WeChat / Weibo Marketing, The Best Low Tax Callers On Board.

Contract Summary

Documentation Quality

Documentation Quality

RAPPER TOKEN provides a document with a good enough standard of solidity base code.

- The technical description is provided clearly and structured, but there are a lot problem with arithmetic operation Issues discovered, state variable visibility is not set, and out of bounds array access

Code Quality

The Overall quality of the basecode is GOOD enough with 30 low-risk issues

- Standart solidity basecode and rules are already followed with RAPPER TOKEN Project .

Test Coverage

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

provides a document with a good standard of solidity base code.

Audit Findings Summary

- SWC-101 | Arithmetic operation Issues discovered on lines 12, 22, 31, 32, 42, 217, 220, 221, 305, 306, 311, 353, 354, 545, 371, 372, 375, 412, 477, 491, 499, 563, 659, 709, 710, 715, 353 and 354.
- SWC-103 | A floating pragma is set on lines 7.
- SWC-108 | State variable visibility is not set on lines 195, 203, 204, 205, 207, 208, 209, 223, 225, 362, 363, 364, 365, 371, 377, 378, 380, 381, 382, 383, 385, 386, 387, 389, 394, 395, 403, 404, and 413. 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.
- SWC-110 | Out of bounds array access on lines 268, 269, 299, 300, 353, 354, 586, 587, 710, 716, 717, and 718.

CONCLUSION

We have audited the RAPPER TOKEN Coin which has released on January 2023 to discover issues and identify potential security vulnerabilities in Goge Project. This process is used to find bugs, technical issues, and security loopholes that find some common issues in the code.

The security audit report produced satisfactory results with a low-risk issue on the contract project.

The most common issue found in writing code on contracts that do not pose a big risk, writing on contracts is close to the standard of writing contracts in general. Some of the low issues that were found were asserting violation and floating pragma is set, we recommend the index access expression can cause an exception in case of use of 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.	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
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Check-Effect Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	PASS
Assert Violation	SWC-110	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 Caller	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
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
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	Thu Jan 19 2023 04:03:42 GMT+0000 (Coordinated Universal Time)
Finished	Fri Jan 20 2023 06:03:49 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	RAPPERTOKEN.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

LINE 12

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
11 library SafeMath {
12 function add(uint256 a, uint256 b) internal pure returns (uint256) {
13 uint256 c = a + b;
14 require(c >= a, "SafeMath: addition overflow");
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 22

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
21  require(b <= a, errorMessage);  
22  uint256 c = a - b;  
23  return c;  
24  |
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 31

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
30  }
31  uint256 c = a * b;
32  require(c / a == b, "SafeMath: multiplication overflow");
33  |
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 32

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
31  uint256 c = a * b;  
32  require(c / a == b, "SafeMath: multiplication overflow");  
33  return c;  
34  |
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 42

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
41   require(b > 0, errorMessage);
42   uint256 c = a / b;
43   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
44   |
```


SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 217

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
216 uint256 public dividendsPerShare;  
217 uint256 public dividendsPerShareAccuracyFactor = 10 ** 36;  
218 //SETMEUP, change this to 1 hour instead of 10mins  
219 |
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 220

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
219 //SETMEUP, change this to 1 hour instead of 10mins
220 uint256 public minPeriod = 30 * 60;
221 uint256 public minDistribution = 1 * (10 ** 12);
222 |
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 221

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
220 uint256 public minPeriod = 30 * 60;
221 uint256 public minDistribution = 1 * (10 ** 12);
222 uint256 currentIndex;
223 |
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 305

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
304 gasLeft = gasleft();
305 currentIndex++;
306 iterations++;
307 }
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 306

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
305     currentIndex++;
306     iterations++;
307 }
308 }
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 311

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
310     return shareholderClaims[shareholder] + minPeriod < block.timestamp
311     && getUnpaidEarnings(shareholder) > minDistribution;
312     }
313     |
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 353

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
352  shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-1];
353  shareholderIndexes[shareholders[shareholders.length-1]] = shareholderIndexes[shareholder];
354  shareholders.pop();
355  |
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 354

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
353  shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-1];
354  shareholderIndexes[shareholders[shareholders.length-1]] = shareholderIndexes[shareholder];
355  shareholders.pop();
356  }
```


SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 371

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
370  uint8 constant _decimals = 2;
371  uint256 _totalSupply = 1 * 10**9 * (10 ** _decimals);
372  uint256 public _maxTxAmount = _totalSupply * 5 / 100;
373  |
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 372

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
371 uint256 _totalSupply = 1 * 10**9 * (10 ** _decimals);
372 uint256 public _maxTxAmount = _totalSupply * 5 / 100;
373 //max wallet holding of 5%
374 |
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 375

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
374 //max wallet holding of 5%
375 uint256 public _maxWalletToken = ( _totalSupply * 5 ) / 100;
376 mapping (address => uint256) _balances;
377 |
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 412

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
411  bool public swapEnabled = true;
412  uint256 public swapThreshold = _totalSupply * 10 / 10000; // 0.01% of supply
413  bool inSwap;
414  modifier swapping() { inSwap = true; _; inSwap = false; }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 477

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
476     function setMaxWalletPercent(uint256 maxWallPercent) external onlyOwner() {
477         _maxWalletToken = (_totalSupply * maxWallPercent) / 100;
478     }
479     |
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 491

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
490     uint256 heldTokens = balanceOf(recipient);
491     require((heldTokens + amount) <= _maxWalletToken, "Total Holding is currently
limited, you can not buy that much.");}
492     |
493     |
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 499

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
498     require(cooldownTimer[recipient] < block.timestamp, "Please wait for 1min between
two buys");
499     cooldownTimer[recipient] = block.timestamp + cooldownTimerInterval;
500     }
501     |
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 563

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
562     uint256 amountBNB = address(this).balance;
563     payable(marketingFeeReceiver).transfer(amountBNB * amountPercentage / 100);
564 }
565 |
```


SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 659

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
658     feeDenominator = _feeDenominator;  
659     require(totalFee < feeDenominator/4);  
660 }
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 709

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
708   require(addresses.length == tokens.length, "Mismatch between Address and token
count");
709   for(uint i=0; i < addresses.length; i++){
710     SCCC = SCCC + tokens[i];
711   }
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 710

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
709   for(uint i=0; i < addresses.length; i++){  
710     SCCC = SCCC + tokens[i];  
711   }  
712   |
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 715

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
714   require(balanceOf(from) >= SCCC, "Not enough tokens in wallet for airdrop");
715   for(uint i=0; i < addresses.length; i++){
716     _basicTransfer(from,addresses[i],tokens[i]);
717     if(!isDividendExempt[addresses[i]]) {
```

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

LINE 353

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
352 function removeShareholder(address shareholder) internal {
353     shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-
1];
354     shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
355     shareholders.pop();
```

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

LINE 354

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
353  shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-1];
354  shareholderIndexes[shareholders[shareholders.length-1]] = shareholderIndexes[shareholder];
355  shareholders.pop();
356  }
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 7

low SEVERITY

The current pragma Solidity directive is ""^0.7.4"". 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

- RAPPERTOKEN.sol

Locations

```
6 //SPDX-License-Identifier: MIT
7 pragma solidity ^0.7.4;
8 |
9 |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 195

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
194 using SafeMath for uint256;
195 address _token;
196 struct Share {
197 |
```


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 "BUSD" is internal. Other possible visibility settings are public and private.

Source File

- RAPPERTOKEN.sol

Locations

```
202  IBEP20  BUSD = IBEP20(0xe9e7CEA3DedcA5984780Bafc599bd69ADd087D56);  
203  address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;  
204  IDEXRouter router;  
205  |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 204

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
203  IBEP20  BUSD = IBEP20(0xe9e7CEA3DedcA5984780Bafc599bd69ADd087D56);  
204  address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;  
205  IDEXRouter router;  
206  |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 205

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
204 address WBNB = 0xbb4CdB9CBd36B01bD1cBaEaF2De08d9173bc095c;  
205 IDEXRouter router;  
206 address[] shareholders;  
207 |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 207

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
206 IDEXRouter router;  
207 address[] shareholders;  
208 mapping (address => uint256) shareholderIndexes;  
209 mapping (address => uint256) shareholderClaims;
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 208

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
207 address[] shareholders;  
208 mapping (address => uint256) shareholderIndexes;  
209 mapping (address => uint256) shareholderClaims;
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 209

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
208 mapping (address => uint256) shareholderIndexes;  
209 mapping (address => uint256) shareholderClaims;  
210 mapping (address => Share) public shares;  
211 |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 223

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
222 uint256 public minDistribution = 1 * (10 ** 12);
223 uint256 currentIndex;
224 bool initialized;
225 |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 225

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
224 uint256 currentIndex;  
225 bool initialized;  
226 modifier initialization() {  
227     require(!initialized);
```


SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 362

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
361 using SafeMath for uint256;
362 address BUSD = 0xe9e7CEA3DedcA5984780BafC599bD69ADd087D56;
363 address WBNB = 0xbb4CdB9CBd36B01bD1cBaEbf2De08d9173bc095c;
364 address DEAD = 0x00000000000000000000000000000000dEaD;
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 363

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
362 address BUSD = 0xe9e7CEA3DedcA5984780Bafc599bD69ADd087D56;  
363 address WBNB = 0xbb4CdB9CBd36B01bD1cBaEbf2De08d9173bc095c;  
364 address DEAD = 0x00000000000000000000000000000000deEaD;  
365 address ZERO = 0x000000000000000000000000000000000000;
```


SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 371

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
370  uint8 constant _decimals = 2;
371  uint256 _totalSupply = 1 * 10**9 * (10 ** _decimals);
372  uint256 public _maxTxAmount = _totalSupply * 5 / 100;
373  |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 377

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
376 uint256 public _maxWalletToken = ( _totalSupply * 5 ) / 100;
377 mapping (address => uint256) _balances;
378 mapping (address => mapping (address => uint256)) _allowances;
379 |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 378

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

- RAPPERTOKEN.sol

Locations

```
377 mapping (address => uint256) _balances;  
378 mapping (address => mapping (address => uint256)) _allowances;  
379 mapping (address => bool) isFeeExempt;  
380 |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 380

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
379 mapping (address => mapping (address => uint256)) _allowances;  
380 mapping (address => bool) isFeeExempt;  
381 mapping (address => bool) isTxLimitExempt;  
382 mapping (address => bool) isTimelockExempt;
```


SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 381

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
380 mapping (address => bool) isFeeExempt;  
381 mapping (address => bool) isTxLimitExempt;  
382 mapping (address => bool) isTimelockExempt;  
383 mapping (address => bool) isDividendExempt;
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 382

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
381 mapping (address => bool) isTxLimitExempt;  
382 mapping (address => bool) isTimelockExempt;  
383 mapping (address => bool) isDividendExempt;  
384 |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 383

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
382 mapping (address => bool) isTimelockExempt;  
383 mapping (address => bool) isDividendExempt;  
384 uint256 liquidityFee = 1;  
385 |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 385

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
384 mapping (address => bool) isDividendExempt;  
385 uint256 liquidityFee = 1;  
386 uint256 reflectionFee = 1;  
387 uint256 marketingFee = 5;
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 386

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
385  uint256 liquidityFee = 1;
386  uint256 reflectionFee = 1;
387  uint256 marketingFee = 5;
388  uint256 public totalFee = 7;
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 387

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
386 uint256 reflectionFee = 1;
387 uint256 marketingFee = 5;
388 uint256 public totalFee = 7;
389 uint256 feeDenominator = 100;
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 389

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
388 uint256 public totalFee = 7;  
389 uint256 feeDenominator = 100;  
390 address public autoLiquidityReceiver;  
391 |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 394

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
393 address public marketingFeeReceiver;  
394 uint256 targetLiquidity = 20;  
395 uint256 targetLiquidityDenominator = 100;  
396 |
```


SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 395

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
394 uint256 targetLiquidity = 20;
395 uint256 targetLiquidityDenominator = 100;
396 IDEXRouter public router;
397 |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 403

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
402  bool public tradingOpen = true;
403  DividendDistributor distributor;
404  uint256 distributorGas = 500000;
405  |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 404

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
403 DividendDistributor distributor;  
404 uint256 distributorGas = 500000;  
405 // Cooldown & timer functionality  
406 |
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 413

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

- RAPPERTOKEN.sol

Locations

```
412 uint256 public swapThreshold = _totalSupply * 10 / 10000; // 0.01% of supply
413 bool inSwap;
414 modifier swapping() { inSwap = true; _; inSwap = false; }
415 |
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 268

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
267 address[] memory path = new address[](2);
268 path[0] = WBNB;
269 path[1] = address(BUSD);
270 |
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 269

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
268 path[0] = WBNB;  
269 path[1] = address(BUSD);  
270 router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: msg.value}(  
271 |
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 299

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
298     }  
299     if(shouldDistribute(shareholders[currentIndex])){  
300         distributeDividend(shareholders[currentIndex]);  
301     }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 300

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
299     if(shouldDistribute(shareholders[currentIndex])){  
300         distributeDividend(shareholders[currentIndex]);  
301     }  
302     |
```


SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 353

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
352 function removeShareholder(address shareholder) internal {
353     shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-
1];
354     shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
355     shareholders.pop();
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 354

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
353  shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-1];
354  shareholderIndexes[shareholders[shareholders.length-1]] = shareholderIndexes[shareholder];
355  shareholders.pop();
356  }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 586

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
585     address[] memory path = new address[](2);
586     path[0] = address(this);
587     path[1] = WBNB
588     |
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 587

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
586 path[0] = address(this);  
587 path[1] = WBNB;  
588 uint256 balanceBefore = address(this).balance;  
589 |
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 710

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
709   for(uint i=0; i < addresses.length; i++){  
710     SCCC = SCCC + tokens[i];  
711   }  
712   |
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 716

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
715   for(uint i=0; i < addresses.length; i++){  
716     _basicTransfer(from,addresses[i],tokens[i]);  
717     if(!isDividendExempt[addresses[i]]) {  
718       try distributor.setShare(addresses[i], _balances[addresses[i]]) {} catch {}
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 717

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
716     basicTransfer(from,addresses[i],tokens[i]);
717     if(!isDividendExempt[addresses[i]]) {
718         try distributor.setShare(addresses[i], _balances[addresses[i]]) {} catch {}
719     }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 718

low SEVERITY

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

Source File

- RAPPERTOKEN.sol

Locations

```
717     if(!isDividendExempt[addresses[i]]) {  
718         try distributor.setShare(addresses[i], _balances[addresses[i]]) {} catch {}  
719     }  
720 }
```


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.