Recharge

Smart Contract Audit Report

SYSFIXED

14 Jul 2021

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Recharge	RCG	Binance Smart Chain

Addresses

Contract address	0x2d94172436d869c1e3c094bead272508fab0d9e3
Contract deployer address	0x3c2465d88C6546eac6F9aa6f79081Ad874CA2E8b

Project Website

https://www.therecharge.io/

Codebase

https://bscscan.com/address/0x2d94172436d869c1e3c094bead272508fab0d9e3#code

SUMMARY

The Recharge is A decentralized incentive solution that connects electric-charging platforms. The Recharge aims to provide a long-term sustainable decentralized ecosystem that can help maximize participating users' incentives.

Contract Summary

Documentation Quality

Recharge 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 Recharge 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 490.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 293, 311, 332, 359, 360, 379, 380, 402, 403, 521, 522, 523, 523, 524, 524, 533, 535, 536, 537, 559, 562, 564, 565, 567, 579, 587, 595, 607, 608, 617 and 522.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 16, 93, 117 and 144.

CONCLUSION

We have audited the Recharge project released on July 2021 to discover issues and identify potential security vulnerabilities in Recharge Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

The security audit report provides satisfactory results with low-risk issues.

The issues found in the Recharge 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 are arithmetic operation issues, a floating pragma is set, and a state variable visibility is not set. The current pragma Solidity directive is ""^0.8.0"". Specifying a fixed compiler version is recommended 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. 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.

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.	PASS
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

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

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

SMART CONTRACT ANALYSIS

Started	Tuesday Jul 13 2021 09:43:58 GMT+0000 (Coordinated Universal Time)
Finished	Wednesday Jul 14 2021 15:37:16 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	RCG.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 293

low SEVERITY

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

Source File

- RCG.sol

Locations

```
292   require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");
293   _approve(sender, _msgSender(), currentAllowance - amount);
294
295   return true;
296   }
297
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 311

low SEVERITY

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

Source File

- RCG.sol

Locations

```
310   function increaseAllowance(address spender, uint256 addedValue) public virtual
returns (bool) {
311   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
312   return true;
313   }
314
315
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 332

low SEVERITY

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

Source File

- RCG.sol

Locations

```
331   require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below
zero");
332   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
333
334   return true;
335   }
336
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 359

low SEVERITY

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

Source File

- RCG.sol

Locations

```
358   require(senderBalance >= amount, "ERC20: transfer amount exceeds balance");
359   _balances[sender] = senderBalance - amount;
360   _balances[recipient] += amount;
361
362   emit Transfer(sender, recipient, amount);
363
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 360

low SEVERITY

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

Source File

- RCG.sol

Locations

```
359   _balances[sender] = senderBalance - amount;
360   _balances[recipient] += amount;
361
362   emit Transfer(sender, recipient, amount);
363   }
364
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 379

low SEVERITY

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

Source File

- RCG.sol

Locations

```
378
379   _totalSupply += amount;
380   _balances[account] += amount;
381   emit Transfer(address(0), account, amount);
382   }
383
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 380

low SEVERITY

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

Source File

- RCG.sol

Locations

```
379   _totalSupply += amount;
380   _balances[account] += amount;
381   emit Transfer(address(0), account, amount);
382   }
383
384
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 402

low SEVERITY

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

Source File

- RCG.sol

Locations

```
401   require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
402   _balances[account] = accountBalance - amount;
403   _totalSupply -= amount;
404
405   emit Transfer(account, address(0), amount);
406
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 403

low SEVERITY

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

Source File

- RCG.sol

Locations

```
402   _balances[account] = accountBalance - amount;
403   _totalSupply -= amount;
404
405   emit Transfer(account, address(0), amount);
406   }
407
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 521

low SEVERITY

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

Source File

- RCG.sol

Locations

```
520   if(basePercent==0) return 0;
521   uint256 c = value+basePercent;
522   uint256 d = c-1;
523   uint256 roundValue = d/basePercent*basePercent;
524   uint256 cutValue = roundValue*basePercent/10000;
525
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 522

low SEVERITY

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

Source File

- RCG.sol

Locations

```
521   uint256 c = value+basePercent;
522   uint256 d = c-1;
523   uint256 roundValue = d/basePercent*basePercent;
524   uint256 cutValue = roundValue*basePercent/10000;
525   return cutValue;
526
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 523

low SEVERITY

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

Source File

- RCG.sol

Locations

```
522   uint256 d = c-1;
523   uint256 roundValue = d/basePercent*basePercent;
524   uint256 cutValue = roundValue*basePercent/10000;
525   return cutValue;
526   }
527
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 523

low SEVERITY

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

Source File

- RCG.sol

Locations

```
522   uint256 d = c-1;
523   uint256 roundValue = d/basePercent*basePercent;
524   uint256 cutValue = roundValue*basePercent/10000;
525   return cutValue;
526   }
527
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 524

low SEVERITY

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

Source File

- RCG.sol

Locations

```
523   uint256 roundValue = d/basePercent*basePercent;
524   uint256 cutValue = roundValue*basePercent/10000;
525   return cutValue;
526   }
527
528
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 524

low SEVERITY

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

Source File

- RCG.sol

Locations

```
523   uint256 roundValue = d/basePercent*basePercent;
524   uint256 cutValue = roundValue*basePercent/10000;
525   return cutValue;
526   }
527
528
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 533

low SEVERITY

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

Source File

- RCG.sol

Locations

```
532   uint256 tokensToBurn = cut(value);
533   uint256 tokensToTransfer = value-tokensToBurn;
534
535   _balances[msg.sender] = _balances[msg.sender]-value;
536   _balances[to] = _balances[to]+tokensToTransfer;
537
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 535

low SEVERITY

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

Source File

- RCG.sol

Locations

```
534
535   _balances[msg.sender] = _balances[msg.sender]-value;
536   _balances[to] = _balances[to]+tokensToTransfer;
537   _balances[benefitial] = _balances[benefitial]+tokensToBurn;
538
539
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 536

low SEVERITY

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

Source File

- RCG.sol

Locations

```
535   _balances[msg.sender] = _balances[msg.sender]-value;
536   _balances[to] = _balances[to]+tokensToTransfer;
537   _balances[benefitial] = _balances[benefitial]+tokensToBurn;
538
539   emit Transfer(msg.sender, to, tokensToTransfer);
540
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 537

low SEVERITY

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

Source File

- RCG.sol

Locations

```
536   _balances[to] = _balances[to]+tokensToTransfer;
537   _balances[benefitial] = _balances[benefitial]+tokensToBurn;
538
539   emit Transfer(msg.sender, to, tokensToTransfer);
540   emit Transfer(msg.sender, benefitial, tokensToBurn);
541
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 559

low SEVERITY

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

Source File

- RCG.sol

Locations

```
558
559   _balances[from] = _balances[from]-value;
560
561   uint256 tokensToBurn = cut(value);
562   uint256 tokensToTransfer = value-tokensToBurn;
563
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 562

low SEVERITY

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

Source File

- RCG.sol

Locations

```
561   uint256 tokensToBurn = cut(value);
562   uint256 tokensToTransfer = value-tokensToBurn;
563
564   _balances[to] = _balances[to]+tokensToTransfer;
565   _balances[benefitial] = _balances[benefitial]+tokensToBurn;
566
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 564

low SEVERITY

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

Source File

- RCG.sol

Locations

```
563
564   _balances[to] = _balances[to]+tokensToTransfer;
565   _balances[benefitial] = _balances[benefitial]+tokensToBurn;
566
567   _allowed[from][msg.sender] = _allowed[from][msg.sender]-value;
568
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 565

low SEVERITY

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

Source File

- RCG.sol

Locations

```
564   _balances[to] = _balances[to]+tokensToTransfer;
565   _balances[benefitial] = _balances[benefitial]+tokensToBurn;
566
567   _allowed[from][msg.sender] = _allowed[from][msg.sender]-value;
568
569
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 567

low SEVERITY

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

Source File

- RCG.sol

Locations

```
566
567   _allowed[from][msg.sender] = _allowed[from][msg.sender]-value;
568
569   emit Approval(from, msg.sender, _allowed[from][msg.sender]);
570   emit Transfer(from, to, tokensToTransfer);
571
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 579

low SEVERITY

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

Source File

- RCG.sol

Locations

```
578   require(spender != address(0), "Address cannot be 0x0");
579   _allowed[msg.sender][spender] = (_allowed[msg.sender][spender]+addedValue);
580   emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
581   return true;
582   }
583
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 587

low SEVERITY

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

Source File

- RCG.sol

Locations

```
586   require(spender != address(0), "Address cannot be 0x0");
587   _allowed[msg.sender][spender] = (_allowed[msg.sender][spender]-subtractedValue);
588   emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
589   return true;
590   }
591
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 595

low SEVERITY

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

Source File

- RCG.sol

Locations

```
594   require(amount != 0, "Amount cannot be 0");
595   _balances[account] = _balances[account]+amount;
596   emit Transfer(address(0), account, amount);
597   }
598
599
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 607

low SEVERITY

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

Source File

- RCG.sol

Locations

```
606   require(amount != 0, "Amount Cannot be 0");
607   _balances[account] = _balances[account]-amount;
608   _totalSupply = _totalSupply-amount;
609   emit Transfer(account, address(0), amount);
610   }
611
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 608

low SEVERITY

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

Source File

- RCG.sol

Locations

```
607   _balances[account] = _balances[account]-amount;
608   _totalSupply = _totalSupply-amount;
609   emit Transfer(account, address(0), amount);
610   }
611
612
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 617

low SEVERITY

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

Source File

- RCG.sol

Locations

```
616   function destroyFrom(address account, uint256 amount) external {
617   _allowed[account][msg.sender] = _allowed[account][msg.sender]-amount;
618   _destroy(account, amount);
619
620   emit Approval(account, msg.sender, _allowed[account][msg.sender]);
621
```

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

LINE 522

low SEVERITY

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

Source File

- RCG.sol

Locations

```
521   uint256 c = value+basePercent;
522   uint256 d = c-1;
523   uint256 roundValue = d/basePercent*basePercent;
524   uint256 cutValue = roundValue*basePercent/10000;
525   return cutValue;
526
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 16

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". 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

- RCG.sol

Locations

```
15
16  pragma solidity ^0.8.0;
17
18  /**
19  * @dev Interface of the ERC20 standard as defined in the EIP.
20
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 93

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". 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

- RCG.sol

Locations

```
92
93   pragma solidity ^0.8.0;
94
95   /*
96   * @dev Provides information about the current execution context, including the
97
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 117

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". 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

- RCG.sol

Locations

```
116
117   pragma solidity ^0.8.0;
118
119
120   /**
121
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 144

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". 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

- RCG.sol

Locations

```
143
144   pragma solidity ^0.8.0;
145
146
147
148
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 490

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

- RCG.sol

Locations

```
489   string public constant tokenSymbol = "RCG";
490   uint256 _totalSupply = 0;
491   uint256 public basePercent = 0;
492
493   constructor(address _owner, uint256 amount) ERC20(tokenName, tokenSymbol)
Owned(_owner) {
494
```

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.