INTRODUCTION

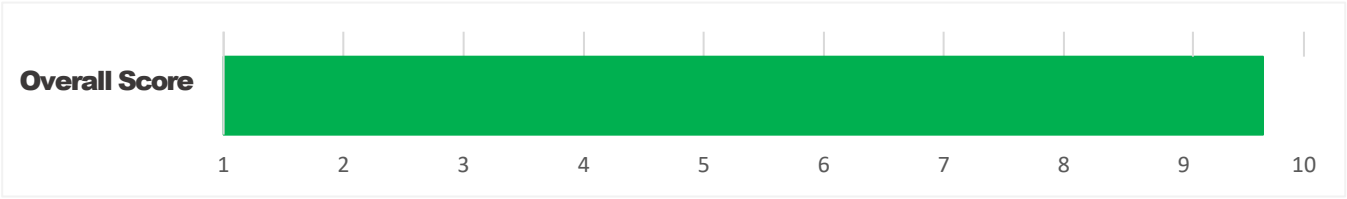
Client Firm	**REDMEMECOIN**
Methodology	**Automated Analysis, Manual Code Review**
Language	**Solidity**
Contract	0x5Ba145C4E526289b1DA530d0c21D5147c9519b41
Blockchain	**Binance Smart Chain**
Centralization	**Active ownership**
Website	**https://redmemecoin.pro/**
Telegram	**https://t.me/RedMemecoin**
Twitter	**https://twitter.com/RedMemeCoin**

EXECUTIVE SUMMARY

performed the automated and manual analysis of the Sol code. The code was reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

Status	Critical ! 🔴	Major " 🟠	Medium # 🟡	Minor $ 🟢	Unknown % 🟤
Open	0	0	0	3	0
Acknowledged	0	0	1	2	0
Resolved	0	0	0	0	0
Noteworty onlyOwner Privileges	Set Taxes and Ratios, Airdrop, Set Protection Settings, Set Reward Properties, Set Reflector Settings, Set Swap Settings, Set Pair and Router				

RedMeMe Smart contract has achieved the following score: 96.8



ℹ️ **Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.**

ℹ️ **Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.**

TABLE OF CONTENTS

TABLE OF CONTENTS 4

SCOPE OF WORK 5

AUDIT METHODOLOGY 6

RISK CATEGORIES 8

CENTRALIZED PRIVILEGES 9

AUTOMATED ANALYSIS 10

INHERITANCE GRAPH 15

SCOPE OF WORK

REDMEMECOIN to conduct the smart contract audit of its .Sol source code. The audit scope of work is strictly limited to mentioned MOVE fileonly:

- **REDMEMECOIN.Sol**

External contracts and/or interfaces dependencies are not checked due to being out of scope.

Verify audited contract's contract address and deployed link below:

Public Contract Link	
0x5Ba145C4E526289b1DA530d0c21D5147c9519b41	
Contract Name	**REDMEMECOIN**
Token Symbol	**REDMEME**
Total Supply	**69,000,000,000**

AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit.

auditing process and methodology:

CONNECT

- **The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.**

AUDIT

- **Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:**
  - **Remix IDE Developer Tool**
  - **Open Zeppelin Code Analyzer**
  - **SWC Vulnerabilities Registry**
  - **DEX Dependencies, e.g., Pancakeswap, Uniswap**
- **Simulations are performed to identify centralized exploits causing contract and/or trade locks.**
- **A manual line-by-line analysis is performed to identify contract issues and centralized privileges. We may inspect below mentioned common contract vulnerabilities, and centralized exploits:**

Centralized Exploits	○ **Token Supply Manipulation** ○ **Access Control and Authorization** ○ **Assets Manipulation** ○ **Ownership Control** ○ **Liquidity Access** ○ **Stop and Pause Trading** ○ **Ownable Library Verification**

Common Contract Vulnerabilities	○ Integer Overflow ○ Lack of Arbitrary limits ○ Incorrect Inheritance Order ○ Typographical Errors ○ Requirement Violation ○ Gas Optimization ○ Coding Style Violations ○ Re-entrancy ○ Third-Party Dependencies ○ Potential Sandwich Attacks ○ Irrelevant Codes ○ Divide before multiply ○ Conformance to Solidity Naming Guides ○ Compiler Specific Warnings ○ Language Specific Warnings

REPORT

- The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.
- The client's development team reviews the report and makes amendments to the codes.
- The auditing team provides the final comprehensive report with open and unresolved issues.

PUBLISH

- The client may use the audit report internally or disclose it publicly.

ℹ️ It is important to note that there is no pass or fail in the audit, it is recommended to view the audit as an unbiased assessment of the safety of solidity codes.

RISK CATEGORIES

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized here for the reader to review:

Risk Type	Definition
Critical !	These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
Major "	These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.
Medium #	These risks should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy-related vulnerabilities should be fixed to deter exploits.
Minor $	These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.
Unknown %	These risks pose uncertain severity to the contract or those who interact with it. They should be fixed immediately to mitigate the risk uncertainty.

All statuses which are identified in the audit report are categorized here for the reader to review:

Status Type	Definition
Open	Risks are open.
Acknowledged	Risks are acknowledged, but not fixed.
Resolved	Risks are acknowledged and fixed.

CENTRALIZED PRIVILEGES

Centralization risk is the most common cause of cryptography asset loss. When a smart contract has a privileged role, the risk related to centralization is elevated.

There are some well-intended reasons have privileged roles, such as:

- **Privileged roles can be granted the power to** pause() **the contract in case of an external attack.**
- **Privileged roles can use functions like,** include(), **and** exclude() **to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale and to list on an exchange.**

Authorizing privileged roles to externally-owned-account (EOA) is dangerous. Lately, centralization-related losses are increasing in frequency and magnitude.

- **The client can lower centralization-related risks by implementing below mentioned practices:**
- **Privileged role's private key must be carefully secured to avoid any potential hack.**
- **Privileged role should be shared by multi-signature (multi-sig) wallets.**
- **Authorized privilege can be locked in a contract, user voting, or community DAO can be introduced to unlock the privilege.**
- **Renouncing the contract ownership, and privileged roles.**
- **Remove functions with elevated centralization risk.**

ℹ️ **Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.**

AUTOMATED ANALYSIS

Symbol	Definition
🛑	**Function modifies state**
#️⃣	**Function is payable**
🔒	**Function is internal**
🔐	**Function is private**
❗	**Function is important**

REDMEMECOIN	Interface			
└	totalSupply	External ❗	❗	NO❗
└	decimals	External ❗	❗	NO❗
└	symbol	External ❗	❗	NO❗
└	name	External ❗	❗	NO❗
└	getOwner	External ❗		NO❗
└	balanceOf	External ❗	❗	NO❗
└	transfer	External ❗	" ❗ 🛑	NO❗
└	allowance	External ❗	❗	NO❗
└	approve	External ❗	" ❗ 🛑	NO❗
└	transferFrom	External ❗	" ❗ 🛑	NO❗
IFactoryV2	Interface			
└	getPair	External ❗	❗	NO❗
└	createPair	External ❗	" ❗ 🛑	NO❗
IV2Pair	Interface			
└	factory	External ❗	❗	NO❗
└	getReserves	External ❗	❗	NO❗
└	sync	External ❗	" ❗ 🛑	NO❗

IRouter01	Interface		
└	factory	External			NO	
└	BNB	External			NO	
└	addLiquidityBNB	External			NO	
└	addLiquidity	External			NO	
└	swapExactAPTForTokens	External			NO	
└	getAmountsOut	External			NO	
└	getAmountsIn	External			NO	
IRouter02	Interface	IRouter01	
└	swapExactTokensForAPTSupportingFeeOnTransferTokens	External			NO	
└	swapExactAPTForTokensSupportingFeeOnTransferTokens	External			NO	
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External			NO	
└	swapExactTokensForTokens	External			NO	
Protections	Interface		
└	checkUser	External			NO	
└	setLaunch	External			NO	
└	setLpPair	External			NO	
└	REDMEME	External			NO	
└	removeSniper	External			NO	
Cashier	Interface		
└	setRewardsProperties	External			NO	
└	tally	External			NO	
└	load	External			NO	
└	cashout	External			NO	
└	giveMeWelfarePlease	External			NO	
└	getTotalDistributed	External			NO	
└	getUserInfo	External			NO	
└	getUserRealizedRewards	External			NO	

| └ | getPendingRewards | External ❗ | | ❗ | NO❗ |

| └ | initialize | External ❗ | " | ❗ | 🛑 | NO❗ |

| └ | getCurrentReward | External ❗ | | ❗ | NO❗ |

||||||

| **SOL** | Implementation | **SafeMath** |||

| └ | <Constructor> | Public ❗ | | ❗ | #️ | NO❗ |

| └ | transferOwner | External ❗ | " | ❗ | 🛑 | onlyOwner |

| └ | renounceOwnership | External ❗ | " | ❗ | 🛑 | NO❗ |

| └ | setOperator | Public ❗ | " | ❗ | 🛑 | NO❗ |

| └ | renounceOriginalDeployer | External ❗ | " | ❗ | 🛑 | NO❗ |

| └ | <Receive Ether> | External ❗ | | ❗ | #️ | NO❗ |

| └ | totalSupply | External ❗ | | ❗ | NO❗ |

| └ | decimals | External ❗ | | ❗ | NO❗ |

| └ | symbol | External ❗ | | ❗ | NO❗ |

| └ | name | External ❗ | | ❗ | NO❗ |

| └ | getOwner | External ❗ | | ❗ | NO❗ |

| └ | balanceOf | Public ❗ | | ❗ | NO❗ |

| └ | allowance | External ❗ | | ❗ | NO❗ |

| └ | approve | External ❗ | " | ❗ | 🛑 | NO❗ |

| └ | _approve | Internal $ | "🔒 | 🛑 | |

| └ | approveContractContingency | Public ❗ | " | ❗ | 🛑 | onlyOwner |

| └ | transfer | External ❗ | " | ❗ | 🛑 | NO❗ |

| └ | transferFrom | External ❗ | " | ❗ | 🛑 | NO❗ |

| └ | setNewRouter | External ❗ | " | ❗ | 🛑 | onlyOwner |

| └ | setLpPair | External ❗ | " | ❗ | 🛑 | onlyOwner |

| └ | setInitializers | External ❗ | " | ❗ | 🛑 | onlyOwner |

| └ | isExcludedFromFees | External ❗ | | ❗ | NO❗ |

| └ | isExcludedFromDividends | External ❗ | | ❗ | NO❗ |

| └ | isExcludedFromProtection | External ❗ | | ❗ | NO❗ |

| └ | setDividendExcluded | Public ❗ | ❗" | 🛑 | onlyOwner |

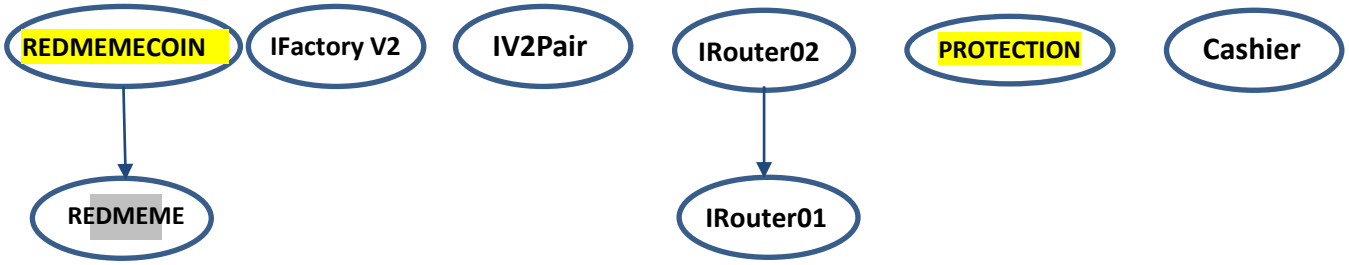
| └ | setExcludedFromFees | Public ❗ | ❗" | 🛑 | onlyOwner |

| └ | getUserRealizedGains | External ❗ | ❗ | NO❗ |

| └ | getUserUnpaidEarnings | External ❗ | ❗ | NO❗ |

| └ | getCurrentReward | External ❗ | ❗ | NO❗ |

INHERITANCE GRAPH



Identifier	Definition	Severity
CEN-12	Centralization privileges of REDMEMECOIN	Medium #

Vulnerability 0 : No important security issue detected.
Threat level: Low

REDMEMECOIN.SOL

C: > Users > Public > REDMEMECOIN.SOL

```
59  // Owned contract
60  // ----------------------------------------------------------------------------
61  contract Owned {
62      address public owner;
63      address public newOwner;
64
65      event OwnershipTransferred(address indexed _from, address indexed _to);
66
67      constructor() public {
68          owner = msg.sender;
69      }
70
71      modifier onlyOwner {
72          require(msg.sender == owner);
73          _;
74      }
75
76      function transferOwnership(address _newOwner) public onlyOwner {
77          newOwner = _newOwner;
78      }
79      function acceptOwnership() public {
80          require(msg.sender == newOwner);
81          emit OwnershipTransferred(owner, newOwner);
82          owner = newOwner;
83          newOwner = address(0);
84      }
85  }
```

ISSUES CHECKING STATUS

	Issue Description	Checking Status
1.	Compiler errors.	PASSED
2.	Race Conditions and reentrancy. Cross-Function Race Conditions.	PASSED
3.	Possible Delay In Data Delivery.	PASSED
4.	Oracle calls.	PASSED
5.	Front Running.	PASSED
6.	Sol. Dependency.	PASSED
7.	Integer Overflow And Underflow.	PASSED
8.	DoS with Revert.	PASSED
9.	Dos With Block Gas Limit.	PASSED
10.	Methods execution permissions.	PASSED
11.	Economy Model of the contract.	PASSED
12.	The Impact Of Exchange Rate On the solidity Logic.	PASSED
13.	Private use data leaks.	PASSED
14.	Malicious Event log.	PASSED
15.	Scoping and Declarations.	PASSED
16.	Uninitialized storage pointers.	PASSED
17.	Arithmetic accuracy.	PASSED
18.	Design Logic.	PASSED
19.	Cross-Function race Conditions	PASSED
20.	Save Upon solidity contract Implementation and Usage.	PASSED
21.	Fallback Function Security	PASSED

AUDIT RESULT

PASSED

SMARTCONTRACT AUDIT OF REDMEMECOIN

Identifier	Definition	Severity
CEN-02	Initial asset distribution	Minor

All of the initially minted assets are sent to the contract deployer when deploying the contract. This can be an issue as the deployer and/or contract owner can distribute tokens without consulting the community.

```
constructor() public {
symbol = "REDMEME";
name = "REDMEMECOIN";
decimals = 9;
_totalSupply = 69000000000 * 10**9;
balances[0xed095292Cc0Bd28CA861e3A05278d903832297F8] = _totalSupply;      emit Transfer(address(0),
```

RECOMMENDATION

Project stakeholders should be consulted during the initial asset distribution process.

RECOMMENDATION

Deployer and/or contract owner private keys are secured carefully.

Please refer to PAGE-09 CENTRALIZED PRIVILEGES for a detailed understanding.

ALLEVIATION

REDMEMECOIN project team understands the centralization risk.Some functions are provided privileged access to ensure a good runtime behaviour in the project

Identifier	Definition	Severity
COD-10	Third Party Dependencies	Minor

Smart contract is interacting with third party protocols e.g., Pancakeswap router, cashier contract, protections contract. The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised, and exploited. Moreover, upgrades in third parties can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

RECOMMENDATION

Inspect and validate third party dependencies regularly, and mitigate severe impacts whenever necessary.