INFINITY

BLOCKCHAIN SOLUTIONS

Smart contract Audit
Full detailed Report

infinityblockchainsolutions.com
contact@infinityblockchainsolutions.com

Table of contents

Contract Review 3
Audit Overview 4
Functions overview 4
Use of Dependencies 6
IBS Risk Analysis 6
Severity Definitions 8
Findings Break Down 8
Centralization 9
Graphical Findings 10
Slither Findings Log 11
Solidity Static Analysis 11
Solhint Linter 17
Final Summary 19
Disclaimer 20
About Infinity Blockchain Solutions 21

This Audit Report was created and compiled by Infinity Blockchain Solutions

Email: contact@infinityblockchainsolutions.com

Contract Review

- Royal is an BEP20-based standard smart contract deployed on the BSC blockchain.

Contract Name	**Royal**
Compiler Version	v0.8.26+commit.8a97fa7a
Optimization	No with 200 runs
Explorer Link	https://bscscan.com/address/0xd0b3e1977738ba02aa47a6bfb9c141d6035dfa79#code
Contract Address	0xD0B3E1977738Ba02aa47a6BFb9C141d6035DFa79
Network	Binance Smart Chain
Symbol	RYL
Decimals	18
Supply	369,000,000 RYL
File Name	Royal.sol
Audit Date	Oct 01, 2025

Audit Overview

This Audit's purpose was to assess any security issues, logic concerns, and potential improvements. After our audit assessment, we have labelled the security state of the contract of Royal to be "**SECURED".**

We have made the use of tools such as Solidity Static analysis, Remix IDE, Slither, Surya along with manual code analysis to inspect the token code.

The details of our findings are presented below.

Functions overview

Sr.	Functions	Type	Observation	Result
1.	constructor	write	Passed	Cleared
2.	name	read	Passed	Cleared
3.	symbol	read	Passed	Cleared
4.	decimals	read	Passed	Cleared
5.	totalSupply	read	Passed	Cleared
6.	transfer	write	Passed	Cleared
7.	allowance	read	Passed	Cleared
8.	approve	write	Passed	Cleared
9.	transferFrom	write	Passed	Cleared
10.	balanceOf	read	Passed	Cleared

11.	_transfer	private	Passed	Cleared
12.	IncreaseAllowance	write	Passed	Cleared
13.	decreaseAllowance	write	Passed	Cleared
14.	_transfer	internal	Passed	Cleared
15.	_mint	internal	Passed	Cleared
16.	_burn	internal	Passed	Cleared
17.	_approve	internal	Passed	Cleared
18.	_spendAllowance	internal	Passed	Cleared
19.	_beforeTokenTransfer	internal	Passed	Cleared
20.	_afterTokenTransfer	internal	Passed	Cleared
21.	onlyOwner	modifier	Passed	Cleared
22.	owner	read	Passed	Cleared
23.	checkOwner	internal	Passed	Cleared
24.	renounceOwnership	write	Access only owner	Cleared
25.	transferOwnership	write	Access only owner	Cleared
26.	_transferOwnership	internal	Passed	Cleared

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are on industry standard libraries like Openzeppelin.

IBS Risk Analysis

Category	Result
Buy Fee	0%
Sell Fee	0%
Cannot Buy	False
Cannot Sell	False
Maximum Tax Cap	False
Tax Modifiable?	Not Detected
Fee Check	Not Detected
Honeypot Issue?	Not Detected
Trading Cooldown	Not Detected
Trade Pausable?	No
Transfer Pausable?	No
Is it Anti-whale?	No
Is Anti-bot?	Not Detected

Can Addresses be Blacklisted?	Not Detected
Blacklist Check	Passed
Mint After deployment?	No
Is it Proxy?	No
Hidden Owner?	Not Detected
Self-Destruction?	Not Detected

Risk Analysis Result: PASSED

This Audit Report was created and compiled by Infinity Blockchain Solutions

Email: contact@infinityblockchainsolutions.com

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities that can lead to token loss etc.
High	Will definitely cause problems; this needs to be adjusted.
Medium	Will likely cause problems and it is recommended to adjust
Low	Won't cause any problems, but can be adjusted for improvement
Informational	Does not compromise the functionality of the contract in any way

Findings Break Down

Risk Level	Unresolved	Acknowledged	Resolved
Critical	0	0	0
High	0	0	0
Medium	0	0	0
Low	0	0	0
Informational	1	1	0

Centralization

Criticality	**Informational**
Status	**Acknowledged**

Description:

This smart contract has some functions which can be executed by the Admin (Owner) only. If the owner wallet private key would be compromised, then it would create trouble.
Following are Owner functions:

Ownable.sol

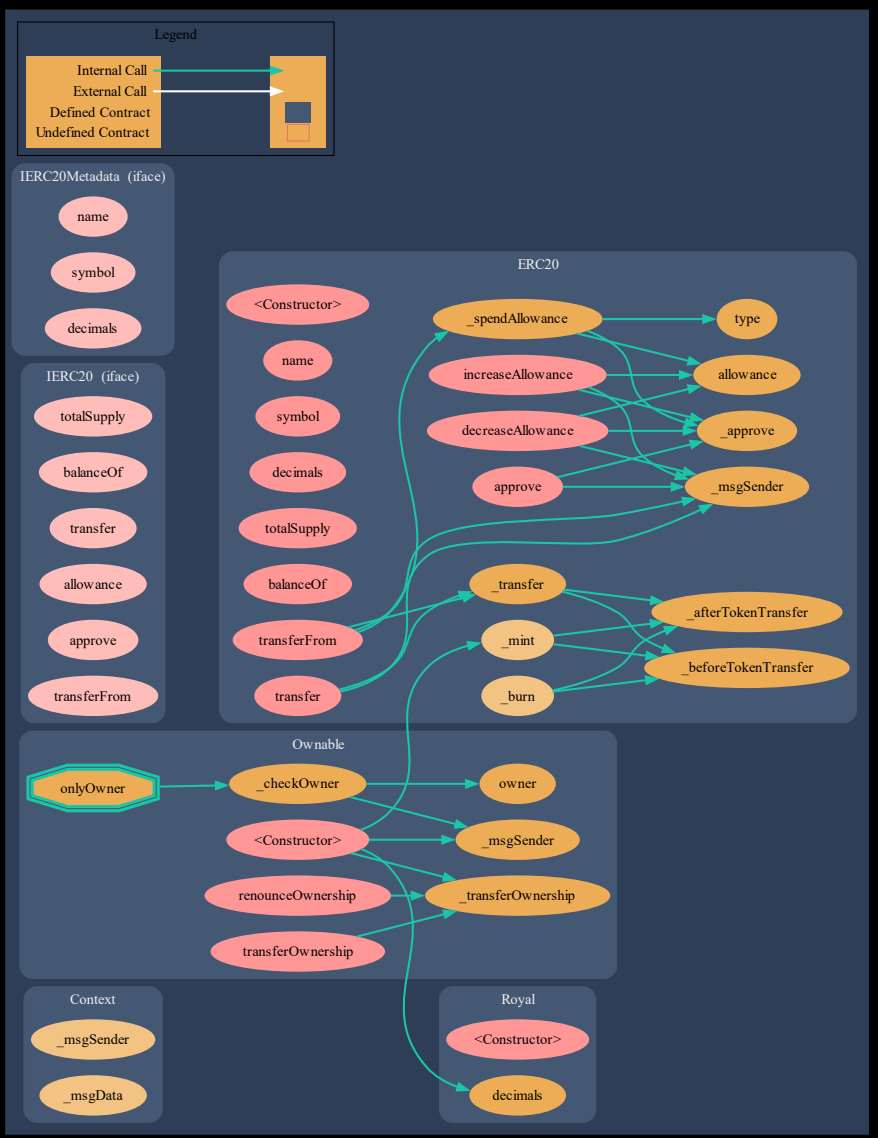
1. renounceOwnership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
2. transferOwnership: The current owner can transfer ownership of the contract to a new account.

Solution

To make the smart contract 100% decentralized, we suggest renouncing ownership of the smart contract once its function is completed.

Graphical Findings

Call Graph Diagram – Royal.sol



Slither Findings Log

Slither is a static analysis tool specifically designed for auditing Solidity smart contracts. It provides fast and comprehensive security analysis, helping developers and auditors identify potential vulnerabilities and bugs in smart contracts before deployment.

Slither Log of Royal.sol:

```
Context._msgData() (contracts/royal.sol#27-29) is never used and should be removed
ERC20._burn(address,uint256) (contracts/royal.sol#536-552) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.26 (contracts/royal.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.26 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Royal.constructor() (contracts/royal.sol#646-648) uses literals with too many digits:
        - _mint(msg.sender,369000000 * 10 ** decimals()) (contracts/royal.sol#647)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (contracts/royal.sol#92-94)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (contracts/royal.sol#100-106)
name() should be declared external:
        - ERC20.name() (contracts/royal.sol#290-292)
symbol() should be declared external:
        - ERC20.symbol() (contracts/royal.sol#298-300)
totalSupply() should be declared external:
        - ERC20.totalSupply() (contracts/royal.sol#322-324)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (contracts/royal.sol#329-333)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (contracts/royal.sol#343-350)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (contracts/royal.sol#372-379)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (contracts/royal.sol#397-406)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (contracts/royal.sol#420-427)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (contracts/royal.sol#443-458)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Solidity Static Analysis

Static code analysis is a technique used to detect common coding issues before the release of a program. It involves reviewing the code either manually or through the use of automated tools. These tools can scan the code without the need for execution, identifying potential problems in advance.

Royal.sol

Gas costs:

Gas requirement of function ERC20.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 290:4:

Gas costs:

Gas requirement of function Royal.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 290:4:

Gas costs:

Gas requirement of function ERC20.symbol is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 298:4:

Constant/View/Pure functions:

IERC20.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 162:4:

Constant/View/Pure functions:

IERC20.approve(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 190:4:

Constant/View/Pure functions:

IERC20.transferFrom(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 201:4:

Similar variable names:

ERC20.(string,string) : Variables have very similar names "_name" and "name_". Note: Modifiers are currently not considered by this static analysis.
Pos: 283:8:

Similar variable names:

ERC20.(string,string) : Variables have very similar names "_name" and "name_". Note: Modifiers are currently not considered by this static analysis.
Pos: 283:16:

Similar variable names:

ERC20.(string,string) : Variables have very similar names "_symbol" and "symbol_". Note: Modifiers are currently not considered by this static analysis.
Pos: 284:8:

Similar variable names:

ERC20.(string,string) : Variables have very similar names "_symbol" and "symbol_". Note: Modifiers are currently not considered by this static analysis.
Pos: 284:18:

No return:

IERC20.balanceOf(address): Defines a return type but never explicitly returns a value.
Pos: 153:4:

No return:

IERC20.transfer(address,uint256): Defines a return type but never explicitly returns a value.
Pos: 162:4:

No return:

IERC20.allowance(address,address): Defines a return type but never explicitly returns a value.
Pos: 171:4:

No return:

IERC20.approve(address,uint256): Defines a return type but never explicitly returns a value.
Pos: 190:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 572:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 573:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

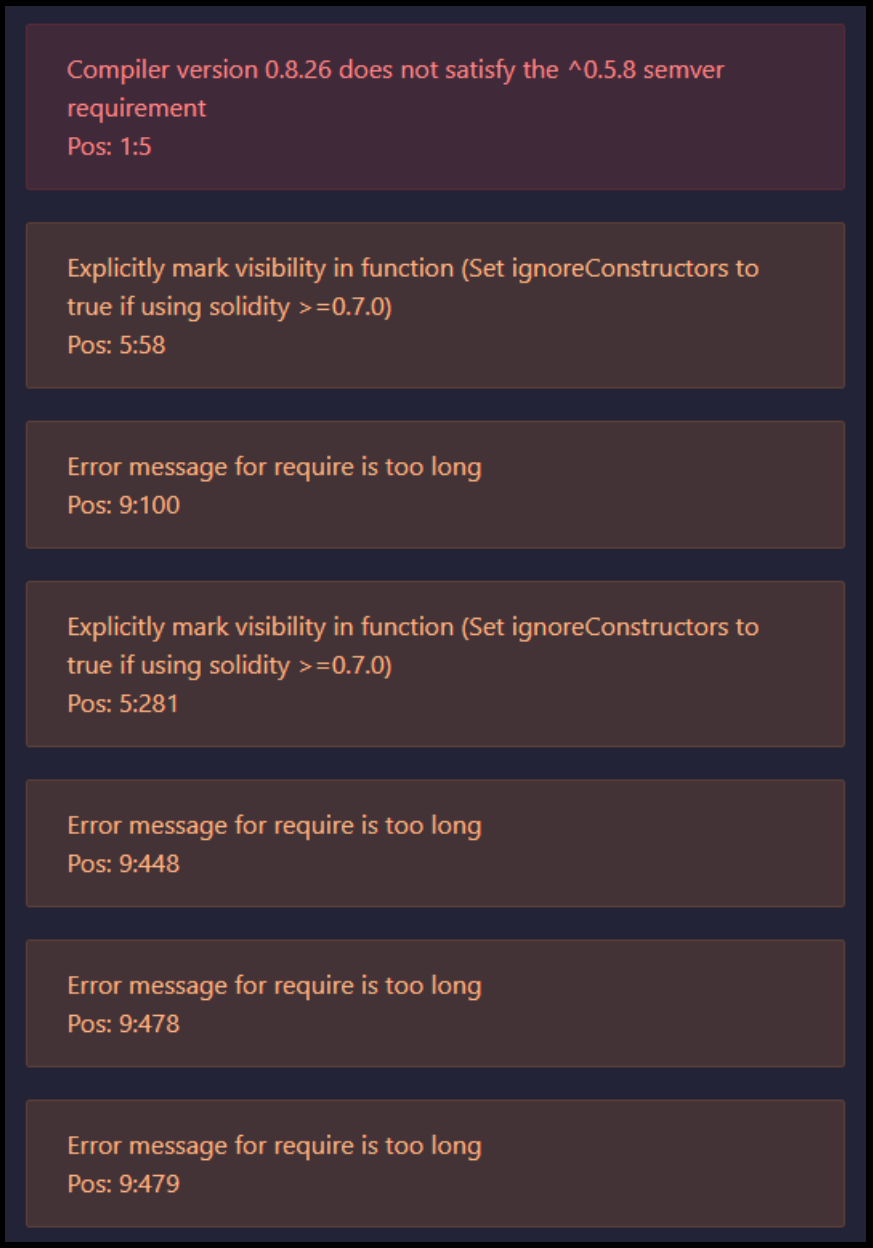
more

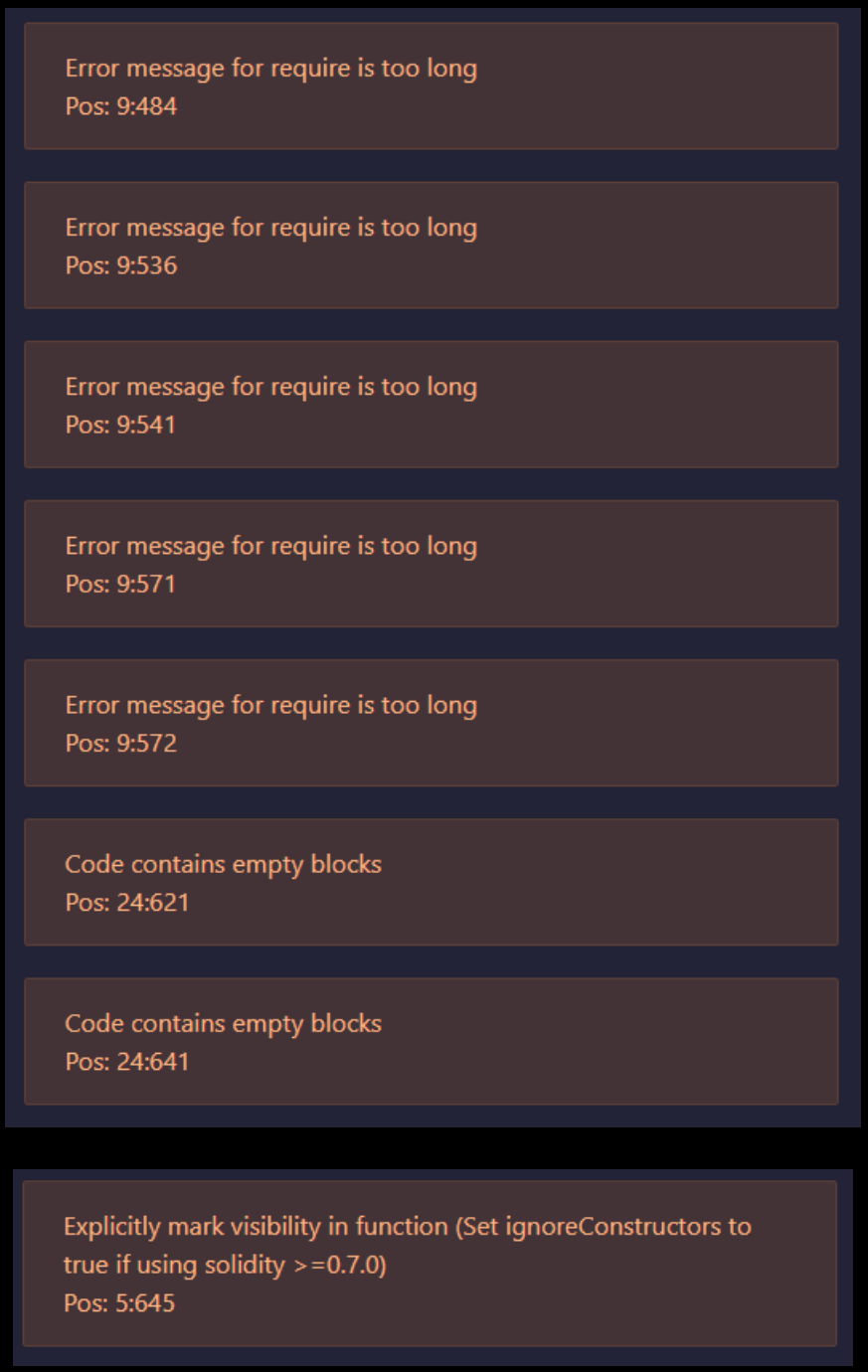
Pos: 594:12:

Solhint Linter

Linters are utility tools designed to analyze source code and identify programming errors, bugs, and stylistic issues.

Royal.sol





Software analysis result:

These tools reported many **false positive results** and some are informational issues.

So, those issues can be **safely ignored**.

Software analysis result: PASSED

Final Summary

This audit investigated any possible issues inside Royal token contract. Our analysis reported no major issues or critical errors. One point to be noted is that the contract owner can access some functions but they cannot be used in a malicious way to disturb user's transactions.

Given that potential test cases for such smart contract protocols can be limitless, we cannot guarantee future outcomes. We have utilized the latest static tools and conducted thorough manual reviews to cover as many test cases as possible.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

As already stated above, we have concluded that the contract's security state is "**SECURED"**.

This Audit Report was created and compiled by Infinity Blockchain Solutions

Email: contact@infinityblockchainsolutions.com

Disclaimer

The information provided in this audit report is based on a thorough analysis conducted by **Infinity Blockchain Solutions**. This audit is intended to evaluate the security, functionality, and best practices of the token's smart contract code.

Our audit is limited to the analysis of the smart contract code provided to us. It does not cover vulnerabilities that may arise from external systems, third-party integrations, or the operational environment in which the token will be deployed.

While we employ industry-standard methodologies and tools, including both manual and automated processes, this audit cannot guarantee the complete absence of vulnerabilities. There may be undiscovered security flaws that were not identified during the review process.

The audit reflects the state of the token's smart contract at the time of review. We cannot predict or protect against future vulnerabilities or exploits that may emerge due to changes in the contract, its environment, or advancements in attack techniques.

Implementing the audit recommendations is the sole responsibility of the project team. Infinity Blockchain Solutions is not liable for any losses or damages resulting from the failure to address issues identified in the audit or any future vulnerabilities that arise post-audit.

This audit does not constitute financial advice. The evaluation is focused solely on the technical aspects of the smart contract. Investors and stakeholders should conduct their own independent research and due diligence before making any financial decisions related to the audited token.

Infinity Blockchain Solutions makes no warranties or representations, either express or implied, regarding the safety, reliability, or security of the audited smart contract. We are not responsible for any loss, damage, or legal issues that may arise from the use or misuse of the token.

About Infinity Blockchain Solutions

Infinity Blockchain Solutions is a leading Web 3.0 development company dedicated to advancing the blockchain ecosystem. Founded with a mission to empower the future of digital innovation, Infinity Blockchain Solutions offers a comprehensive suite of services, including Crypto Token creation, Website development, ICO creation, smart contract audits and more.

With a reputation for excellence and a commitment to security, we have collaborated with numerous projects, contributing to the growth and integrity of the blockchain space. Our expert team provides reliable solutions that ensure the success and safety of our clients' ventures, securing their digital assets and fostering innovation in the decentralized landscape.

INFINITY
BLOCKCHAIN SOLUTIONS

infinityblockchainsolutions.com