ASTROBIATECH
BLOCKCHAIN SECURITY

MADE IN INDIA
BLOCKCHAIN SECURITY

SECURITY ASSESSMENT REPORT

PREPARED FOR
SOULCOIN

@astrobiatech

JULY
2023

astrobiatech.in

TABLE OF CONTENTS

SCOPE OF AUDIT

The scope of this audit was to analyze and document the SoulCoin smart contract codebase for quality, security, and correctness.

CHECKED VULNERABILITIES

We have scanned the smart contract for commonly known and morespecific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

TECHNIQUES & METHODS

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all thesmart contracts.

Static Analysis

Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to testsecurity of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis.Contracts were completely manually analyzed, their logic was checkedand compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

ISSUE CATEGORIES

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

➢ HIGH SEVERITY ISSUES

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before movingto a live environment.

➢ MEDIUM SEVERITY ISSUES

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

➢ LOW SEVERITY ISSUES

Low level severity issues can cause minor impact and or are just warningsthat can remain unfixed for now. It would be better to fix these issues atsome point in the future.

➢ INFORMATIONAL

These are severity four issues which indicate an improvement request, ageneral question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

ISSUES TABLE

TYPE	HIGH	MEDIUM	LOW	INFORMATIONAL
OPEN	2	1	1	1
ACKNOWLWDGENT	-	-	-	-
CLOSED	-	-	-	-

INTRODUCTION

On 24-07-2023 – Astrobiatech Blockchain Security Team performed asecurity audit for SoulCoin smart contracts.

CONTRACT NAME	Soulverse
CONTRACT ADDRESS	0x278E96A2585e65F840bFBacE47CD98E7e88E55bE
BLOCKCHAIN	Polygon (Testnet)
TOTAL SUPPLY	21,000,000,000
SYMBOL	SoulX
DECIMALS	18

OVERVIEW

CONTRACT ADDRESS
0x278E96A2585e65F840bFBacE47CD98E7e88E55bE

TOKEN TRACKER
SoulCoin (SoulX)

CONTRACT CREATOR
0x20626af23194b103da5dd3dbe2334a66aecffe30

OWNER ADDRESS
0x20626af23194b103da5dd3dbe2334a66aecffe30

SOURCE CODE
Contract Source Code Verified at Polygon Testnet

CONTRACT NAME
Soulverse

OTHER SETTINGS
default evmVersion, MIT license

COMPILER VERSION
v0.8.9+commit.e5eed63a

OPTIMIZATION ENABLED
No with 200 runs

Code is truncated to fit the constraints of this document.
https://mumbai.polygonscan.com/token/0x278e96a2585e65f840bfbace47cd98e7e88e55be#code

MANUAL ANALYSIS FINDINGS

HIGH

1. Incorrect Daily Transfer Limit Handling

Description:-

The transfer function in the Soulverse contract has a critical vulnerability related to the daily transfer limit. It fails to update the timestamp for the next allowed transfer correctly, leading to users being unable to transfer tokens even after the time limit has passed.

Code:-

```
function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
require(amount >= MIN_WALLET_HOLDING, "Amount below minimum per wallet");
require(balanceOf(recipient).add(amount) <= MAX_WALLET_HOLDING, "Amount exceeds maximum per wallet");
require(balanceOf(msg.sender).sub(amount) >= MIN_WALLET_HOLDING, "Sender balance will be below minimum per wallet");
require(isWhitelisted(msg.sender), "Sender is not whitelisted");
// Add the validation for maximum selling per wallet per day
require(userTransfers[msg.sender].perDayTransfer.add(amount) <= MAX_SELLING_PER_DAY, "Maximum selling per wallet per day reached");
userTransfers[msg.sender].perDayTransfer = userTransfers[msg.sender].perDayTransfer.add(amount);


return super.transfer(recipient, amount);
}


/**
 * @notice Overrides the transferFrom function to include validation for maximum and minimum token holdings per wallet
 * @param sender The address to transfer tokens from
 * @param recipient The address to transfer tokens to
 * @param amount The amount of tokens to transfer
 * @return A boolean indicating the success of the transfer
 */
function transferFrom(address sender, address recipient, uint256 amount) public virtual override returns (bool) {
    require(amount >= MIN_WALLET_HOLDING, "Amount below minimum per wallet");
    require(balanceOf(recipient).add(amount) <= MAX_WALLET_HOLDING, "Amount exceeds maximum per wallet");
    require(isWhitelisted(msg.sender), "Sender is not whitelisted");
    return super.transferFrom(sender, recipient, amount);
}
}
```

Recommendation:-

Update the transfer function to correctly manage the timestamp for the next allowed transfer in the userTransfers mapping. Implement a mechanism to reset the daily transfer count when the next day begins. Consider applying similar checks for the transferFrom function if required.

2. Honeypot Access Restriction

Description:-

The Soulverse contract restricts token transfers, buying, and selling to only specific whitelisted wallets, implementing a honeypot functionality. This approach contradicts the decentralized and trustless nature of blockchain and smart contracts. It can lead to unintended consequences and is not recommended for most use cases.

Code:-

```
    function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
    require(amount >= MIN_WALLET_HOLDING, "Amount below minimum per wallet");
    require(balanceOf(recipient).add(amount) <= MAX_WALLET_HOLDING, "Amount exceeds maximum per wallet");
    require(balanceOf(msg.sender).sub(amount) >= MIN_WALLET_HOLDING, "Sender balance will be below minimum per wallet");
    require(isWhitelisted(msg.sender), "Sender is not whitelisted");
    // Add the validation for maximum selling per wallet per day
    require(userTransfers[msg.sender].perDayTransfer.add(amount) <= MAX_SELLING_PER_DAY, "Maximum selling per wallet per day reached");
    userTransfers[msg.sender].perDayTransfer = userTransfers[msg.sender].perDayTransfer.add(amount);


    return super.transfer(recipient, amount);
}


    /**
     * @notice Overrides the transferFrom function to include validation for maximum and minimum token holdings per wallet
     * @param sender The address to transfer tokens from
     * @param recipient The address to transfer tokens to
     * @param amount The amount of tokens to transfer
     * @return A boolean indicating the success of the transfer
     */
    function transferFrom(address sender, address recipient, uint256 amount) public virtual override returns (bool) {
        require(amount >= MIN_WALLET_HOLDING, "Amount below minimum per wallet");
        require(balanceOf(recipient).add(amount) <= MAX_WALLET_HOLDING, "Amount exceeds maximum per wallet");
        require(isWhitelisted(msg.sender), "Sender is not whitelisted");
        return super.transferFrom(sender, recipient, amount);
    }
}
```

Recommendation:-

Remove the isWhitelisted modifier and associated checks from the transfer and transferFrom functions. Embrace the standard ERC-20 token behavior to allow any address to transfer tokens freely. If access control is required, consider using OpenZeppelin's access control library or similar mechanisms.

MEDIUM

3. Elevated Privileges with Operator Role

Description:-

The Soulverse contract grants an operator address the same powers as the contract owner through the onlyAuthorized modifier. This operator role can perform all authorized actions, even if the contract owner renounces ownership, potentially leading to centralized control.

Recommendation:-

Reevaluate the need for an operator role. If necessary, limit the operator's scope to specific privileges. Consider implementing a multi-signature mechanism or using an access control library like OpenZeppelin to enhance security and decentralization.

LOW

3. Redundant Function

Description:-

The getOwner function in the Ownable contract is redundant as it provides the same functionality as the owner function. Both functions return the contract owner's address. Having two functions that serve the same purpose can lead to confusion and unnecessary code duplication.

Recommendation:-

Remove the getOwner function as it is redundant. Developers can use the owner function to retrieve the contract owner's address, and there is no need for an additional function with the same functionality.

INFORMATIONAL

4. Deprecated SafeMath Library

Description:-

The SafeMath library used in the contract is explicitly marked as being intended for Solidity versions earlier than 0.8. The contract itself uses Solidity version 0.8.9, which has built-in overflow checking, making the SafeMath library redundant.

Recommendation:-

Since the contract uses Solidity version 0.8.9, it is safe to remove the SafeMath library to reduce unnecessary gas costs and simplify the code. In Solidity 0.8 and above, arithmetic operations have built-in overflow checking by default, making the SafeMath library unnecessary. Consider removing the library from the contract to streamline the code.

AUTOMATED ANALYSIS

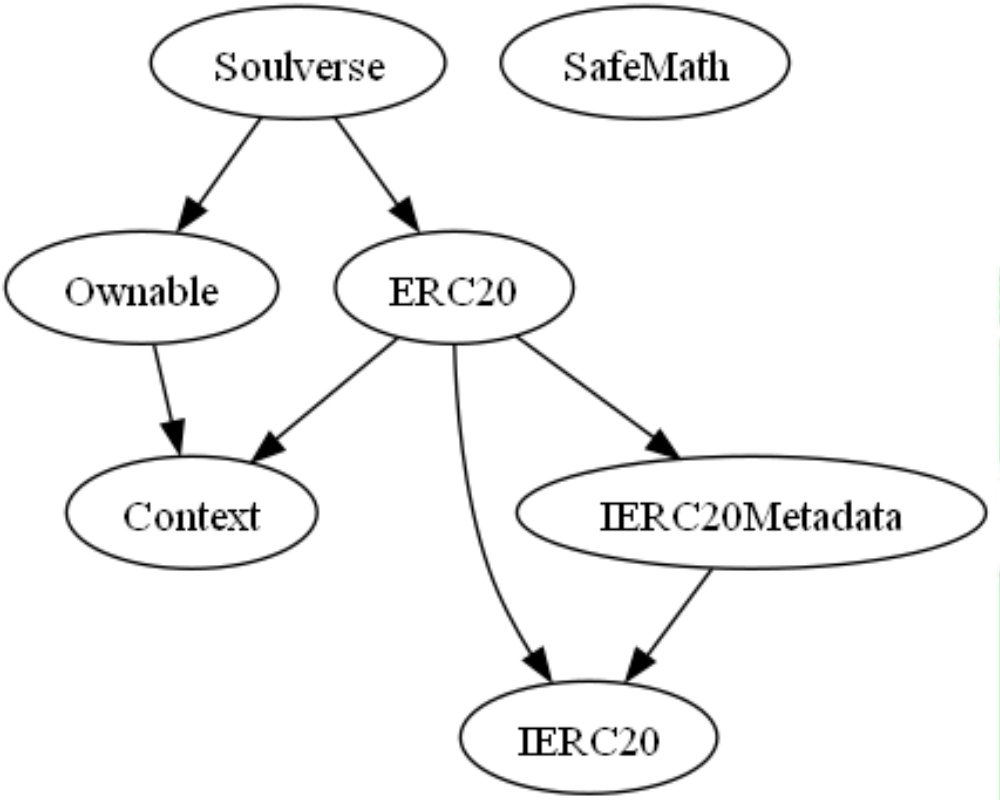
```
Soulverse.setOperator(address)._operator (token.sol#582) lacks a zero-check on :
                - operator = _operator (token.sol#583)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Context._msgData() (token.sol#14-16) is never used and should be removed
SafeMath.div(uint256,uint256) (token.sol#432-434) is never used and should be removed
SafeMath.div(uint256,uint256,string) (token.sol#484-489) is never used and should be removed
SafeMath.mod(uint256,uint256) (token.sol#448-450) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (token.sol#506-511) is never used and should be removed
SafeMath.mul(uint256,uint256) (token.sol#418-420) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (token.sol#465-470) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (token.sol#319-325) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (token.sol#361-366) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (token.sol#373-378) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (token.sol#344-354) is never used and should be removed
SafeMath.trySub(uint256,uint256) (token.sol#332-337) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.9 (token.sol#6) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter Soulverse.setOperator(address)._operator (token.sol#582) is not in mixedCase
Parameter Soulverse.setTransferLimit(uint256)._limit (token.sol#600) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-
conventions
INFO:Detectors:
Soulverse.slitherConstructorConstantVariables() (token.sol#520-666) uses literals with too many digits:
        - FIXED_SUPPLY = 21000000000 * 10 ** 18 (token.sol#523)
Soulverse.slitherConstructorConstantVariables() (token.sol#520-666) uses literals with too many digits:
        - MAX_WALLET_HOLDING = 1000000 * 10 ** 18 (token.sol#529)
Soulverse.slitherConstructorConstantVariables() (token.sol#520-666) uses literals with too many digits:
        - MAX_SELLING_PER_DAY = 100000 * 10 ** 18 (token.sol#531)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

FUNCTIONAL ANALYSIS

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🔴	NO ❗
└	owner	Public ❗		NO ❗
└	getOwner	External ❗		NO ❗
└	transferOwnership	Public ❗	🔴	onlyOwner
└	_transferOwnership	Internal 🔒	🔴	
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🔴	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🔴	NO ❗
└	transferFrom	External ❗	🔴	NO ❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<Constructor>	Public ❗	🔴	NO ❗
└	name	Public ❗		NO ❗
└	symbol	Public ❗		NO ❗
└	decimals	Public ❗		NO ❗
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	transfer	Public ❗	🔴	NO ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🔴	NO ❗
└	transferFrom	Public ❗	🔴	NO ❗
└	increaseAllowance	Public ❗	🔴	NO ❗


└	decreaseAllowance	Public !	🔴	NO !
└	_transfer	Internal 🔒	🔴	
└	_mint	Internal 🔒	🔴	
└	_burn	Internal 🔒	🔴	
└	_approve	Internal 🔒	🔴	
└	_spendAllowance	Internal 🔒	🔴	
└	_beforeTokenTransfer	Internal 🔒	🔴	
└	_afterTokenTransfer	Internal 🔒	🔴	
SafeMath	Library			
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
Soulverse	Implementation	ERC20, Ownable		
└	<Constructor>	Public !	🔴	ERC20
└	setOperator	External !	🔴	onlyOwner
└	burn	External !	🔴	NO !
└	setTransferLimit	External !	🔴	onlyAuthorized
└	whitelistAccount	External !	🔴	onlyAuthorized
└	unWhitelistAccount	External !	🔴	onlyAuthorized
└	isWhitelisted	Public !		NO !
└	blacklistAccount	External !	🔴	onlyAuthorized
└	unBlacklistAccount	External !	🔴	onlyAuthorized
└	isBlacklisted	Public !		NO !
└	transfer	Public !	🔴	NO !
└	transferFrom	Public !	🔴	NO !

INHERITANCE TREE



HOLDERS

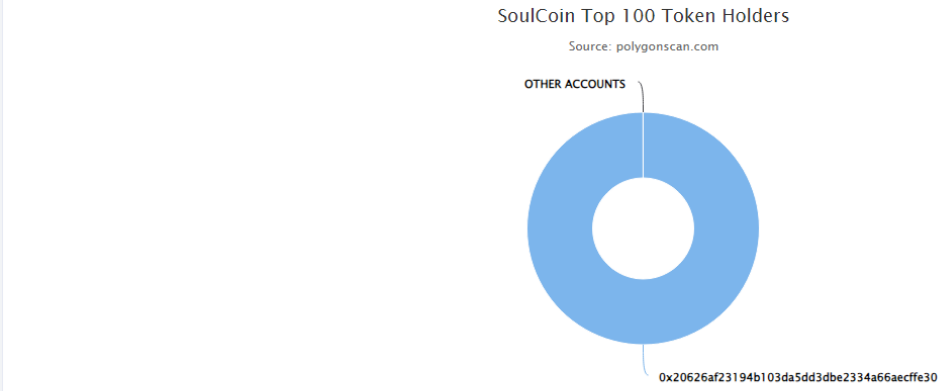
A total of 1 token holder

First | < | Page 1 of 1 | > | Last

Rank	Address	Quantity	Percentage	Analytics
1	0x20626af23194b103da5dd3dbe2334a66aecffe30	21,000,000,000	100.0000%	

The top 100 holders collectively own 100.00% (21,000,000,000.00 Tokens) of SoulCoin

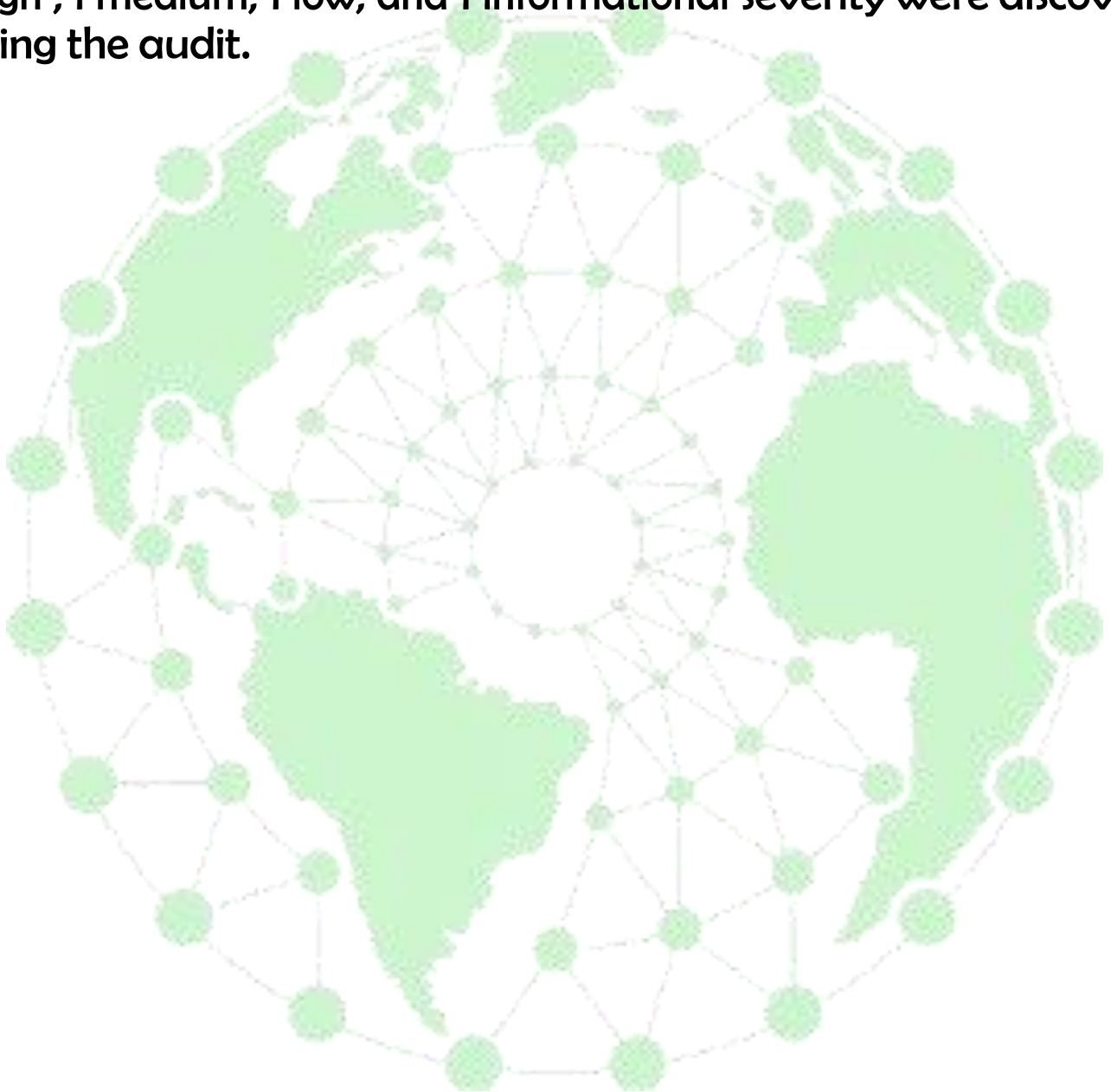
Token Total Supply: 21,000,000,000.00 Token | Total Token Holders: 1



(A total of 21,000,000,000.00 tokens held by the top 100 accounts from the total supply of 21,000,000,000.00 token)

SUMMARY

In this report, we have considered the security of the SoulCoinplatform. We performed our audit according to the procedure described above. 2 high , 1 medium, 1 low, and 1 informational severity were discovered during the audit.



DISCLAIMER

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constituteinvestment advice. No one shall have any right to rely on the report or its contents, and Astrobiatech Blockchain Security and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Astrobiatech BlockchainSecurity) owe no duty of care towards you or any other person, nor does Astrobiatech Blockchain Security make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Astrobiatech Blockchain Security hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effectin relation to the report. Except and only to the extent that it is prohibited by law, Astrobiatech Blockchain Security hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Astrobiatech Blockchain Security, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic lossor damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

SOULCOIN

Proofed by Astrobiatech

ASTROBIATECH
BLOCKCHAIN SECURITY

https://astrobiatech.in

@astrobiatech