QuillAudits

Audit Report
November, 2023

For

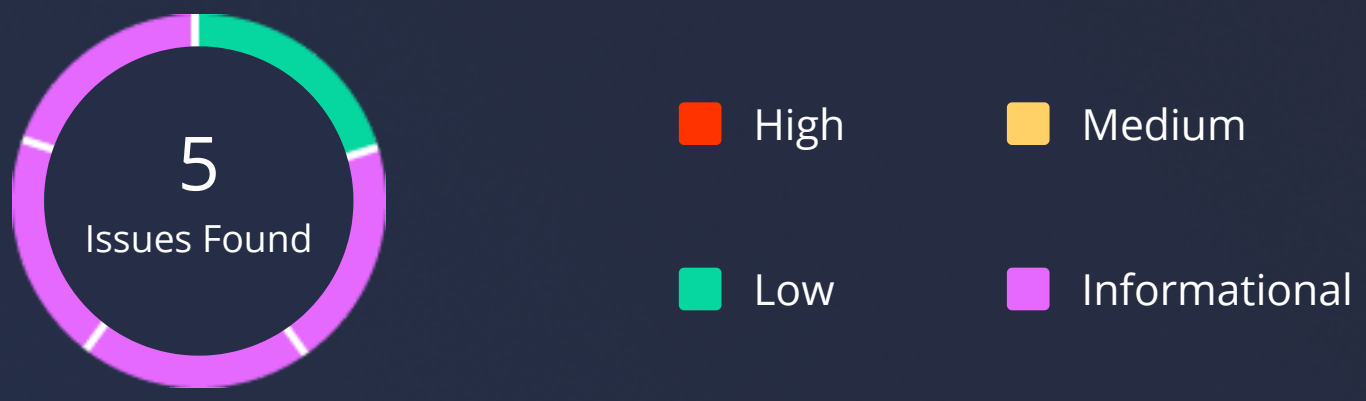
SOULVERSE
POWERED BY SOUL ID™

Table of Content

Executive Summary

Project Name	Soulverse
Project URL	*https://soulverse.us*
Overview	Soulverse is an utility ERC20 token contract that mints a fixed supply to the contract owner when it is deployed. The fixed supply is 21,000,000,000. The contract is characterized for its additional features which include the maximum amount a wallet can hold, the daily transfer limit, and the exception of transfer limit on whitelisted addresses. The contract is designed to authorize the contract owner to whitelist and blacklist addresses.
Audit Scope	*https://mumbai.polygonscan.com/address/0x7c34910F576fA87f7A728A1c5BD4573E2d9913Ec#code*
Contracts in Scope	Soulverse.sol
Commit Hash	NA
Language	Solidity
Blockchain	Polygon
Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	2nd November 2023 - 8th November 2023
Updated Code Received	15th November 2023
Review 2	15th November 2023 - 17th November 2023
Fixed In	*https://polygonscan.com/address/0x456881071305f778ee90c12fc406492691B00603#code*

Number of Security Issues per Severity



	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	1	4

Checked Vulnerabilities

- Access Management
- Arbitrary write to storage
- Centralization of control
- Ether theft
- Improper or missing events
- Logical issues and flaws
- Arithmetic Correctness
- Race conditions/front running
- SWC Registry
- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Malicious libraries
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- ERC's conformance
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Multiple Sends
- Using suicide
- Using delegatecall
- Upgradeable safety
- Using throw

Checked Vulnerabilities

- Using inline assembly
- Unsafe type inference
- Style guide violation
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts, 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 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 the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

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

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Solhint, Mythril, Slither, Solidity Statistic Analysis.

Types of Severity

Every issue in this report has been assigned to 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 be fixed before moving to 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 are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

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

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

A. Contract - Soulverse

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

A.1: Missing Unit Test Cases

Description

There are no present unit test cases. Test coverage is important to show areas in the contract that have been thoroughly tested.

Remediation

Add unit test cases.

Status

Resolved

Informational Issues

A.2: Emit event for State Changes

```
637     function setMaxTokenHoldLimit(uint256 limit) external onlyAuthorized {
638         maxTokenHoldLimit = limit;
639     }
640
```

Description

The setMaxTokenHoldLimit function is a privileged function that only the contract owner can invoke. However, when this function is called to set the maximum token amount that a user wallet can possess, there's no event emitting the changes to ease data tracking.

Remediation

Add an event for the setMaxTokenHoldLimit function.

Status

Resolved

A.3: Remove Unused Code

```
642
643     // function setOperator(address _operator) external onlyOwner {
644     //     operator = _operator;
645     //     emit SetOperator(_operator);
646     // }
647
```

```
611     event TransferLimitSet(uint256 indexed limit);
612
613     // event SetOperator(address operator);
614
```

Description

In the contract, there was a commented implementation for setting of the operator. The operator address was as well commented out, likewise the event created for it. To derive a clean codebase, remove this commented piece of code.

Remediation

Remove unused commented code implementation in the contract.

Status

Resolved

A.4: Inconsistency in Code Implementation and Comment

```
582 contract Soulverse is ERC20, Ownable {
583     // Fixed supply, 2160B tokens
584     uint256 public constant FIXED_SUPPLY = 21000000000 * 10**18;
585
```

Description

In the whitepaper, the total supply of the Soulcoin is 21 billion tokens. This is what was implemented in the code itself but the comment explaining the state variable, “Fixed supply, 2160B tokens”, says otherwise. It’s important that the comment correlates the accurate code implementation to avoid any misunderstanding.

Remediation

Correct the comment to correlate with the exact code.

Status

Resolved

A.5: Add Proper Code Comment Across All Functions

Description

Unlike setTransferLimit and burn function, other functions in the contract are not duly commented. The benefit of this comment is the ability to convey the motive behind a function and explain what and which parameters it accepts. It assists in the comprehension of the contract flow in general.

Remediation

Proper code comments across all the functions would help understand the motives behind functions without any comment.

Status

Resolved

General Recommendation

Soulcoin is a utility token for the Soulverse community. For this purpose, the contract has implementation that limits how much transfer an address can make in a day, the maximum amount that an address can hold and also gives the contract admin the privilege of whitelisting, unwhitelisting, blacklisting and unblacklisting of accounts. When addresses are whitelisted, it skips the daily transfer limit for these addresses and vis-visa. When an address is blacklisted, it cannot receive tokens or be able to transfer its tokens.

Functional Tests

Some of the tests performed are mentioned below:

- ✓ Should get the name of the token
- ✓ Should get the symbol of the token
- ✓ Should get the decimal of the token
- ✓ Should get the total supply of the token when deployed
- ✓ Should get balance of the owner when contract is deployed
- ✓ Should transfer tokens to other address
- ✓ Should approve another account to spend token
- ✓ Should access the balance of allowance allowed to an address
- ✓ Should increase the balance of allowance allowed to an address
- ✓ Should decrease the balance of allowance allowed to an address
- ✓ Should reverts when users attempt to transfer beyond the set transfer limit figure
- ✓ Should successfully transfer to an address and revert when it reaches the maximum token limit to hold
- ✓ Should skip daily limit check when either the from or the to address is whitelisted
- ✓ Should revert transaction from blacklisted addresses

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of Soulverse. We performed our audit according to the procedure described above.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Soulverse smart contracts. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of Soulverse smart contracts. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the Soulverse to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

850+
Audits Completed

$30B
Secured

$30B
Lines of Code Audited

Follow Our Journey

Audit Report
November, 2023

For

SOULVERSE
POWERED BY SOUL ID™

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com