QuillAudits

Audit Report
June, 2022

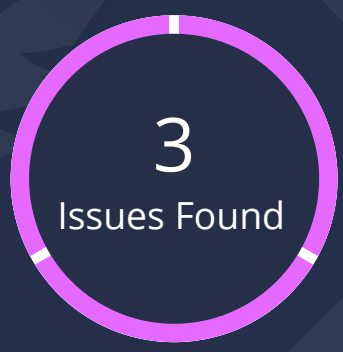
For

SUREDCU
Suriname Reserve Digital Currency

Table of Content

Executive Summary

Project Name	SureDCU
Overview	“Suriname Reserve Digital Currency”, with the Foundation trade name “SUREDCU” and ticker “SRDC”, is an Ethereum-based Cryptocurrency which provides users in the LAT-AM region with the safest and easiest way to onboard into the DeFi (Decentralized Finance) world, so they can exchange other Cryptocurrencies or ALTcoins, without the need of having a bank-account or credit-card. The SRDC uses the same parameters of traditional finance, so users get acquainted with something they already know, but at the same time creating a bridge between traditional finance and the new way in banking, making a much smoother transition for these users into the sophisticated DeFi world.
Timeline	13 May, 2022 to 31 May, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse SureDCU codebase for quality, security, and correctness.
Sourcecode	*https://etherscan.io/address/0xb6244cefb7dc2e42c41bf412a48baf47ca81e084#code*



High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	3
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	0	0

Types of Severities

High

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

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

Low-level severity issues can cause minor impact and or 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 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.

Checked Vulnerabilities

- 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 leve

Techniques and 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 the smart contracts.

Structural Analysis

In this step, we have analysed 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

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 analysed, 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 behaviour 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, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Analysis

A. Contract - TeamToken.sol

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

No issues were found

Informational Issues

A.1 Unlocked pragma (pragma solidity >=0.6.2 <0.8.0)

Description

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation

Here all the in-scope contracts have an unlocked pragma, it is recommended to lock the same. Moreover, we strongly suggest not to use experimental Solidity features (e.g., pragma experimental ABIEncoderV2) or third-party unaudited libraries. If necessary, refactor the current code base to only use stable features.

Status

Acknowledged

A.2 General Recommendations

Front Running

The ERC20 token standard includes a function called 'approve' which allows an address to approve another address to spend tokens on their behalf. The miner who solves the block also chooses which transactions from the pool will be included in the block, typically ordered by the gasPrice of each transaction.
An attacker can watch the transaction pool for transactions that may contain solutions to problems, and modify or revoke the solver's permissions or change state in a contract detrimentally to the solver. The attacker can then get the data from this transaction and create a transaction of their own with a higher gasPrice so their transaction is included in a block before the original.

Remediation

Away to mitigate the threat is to approve token transfers only to smart contracts with verified source code that does not contain logic for performing attacks, and to accounts owned by the people you may trust. Although there are multiple workarounds for this vulnerability. For more details visit the following link - *https://rb.gy/xyxshw*

Use of SafeMath

In our conclusion we have observed that the "SafeMath" library was imported in the contracts and there is a mathematical calculation in the contract on line #30 and #31. We recommend using safeMath library to carry out the operation in order to avoid and potential risk of any arithmetic errors in the future.

Status

Acknowledged

B. Contract - MintBurnTeamToken.sol

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

No issues were found

Informational Issues

B.1 Unlocked pragma (pragma solidity >=0.6.2 <0.8.0)

Description

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation

Here all the in-scope contracts have an unlocked pragma, it is recommended to lock the same. Moreover, we strongly suggest not to use experimental Solidity features (e.g., pragma experimental ABIEncoderV2) or third-party unaudited libraries. If necessary, refactor the current code base to only use stable features.

Status

Acknowledged

Functional Testing

Some of the tests performed are mentioned below

- Should be able to mint tokens
- Should be able to transfer ownership
- Should be able to setup decimals while minting
- Should be able to transfer tokens
- Should be able to deduct a 5% fee and transfer it to the developers while minting
- Should revert if the address of the owner(minter) is not valid
- Should revert if the wallet address of the developer is not valid
- Should revert if total supply is less than or equal to 0
- Should revert if the decimals entered for the token are less than 8 and greater than 18

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 the SureDCU. We performed our audit according to the procedure described above.

Some issues of Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture. At the end, SureDCU team acknowledged all issues.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the SureDCU Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the SureDCU Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

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.

500+
Audits Completed

$15B
Secured

500K
Lines of Code Audited

Follow Our Journey

Audit Report
June, 2022

For

SUREDCU
Suriname Reserve Digital Currency

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com