Centric Protocol Smart Contract Audit

Centric Foundation, 07 April 2020

1. Introduction

iosiro was commissioned by the Centric Foundation to conduct a smart contract audit on the Centric Rise and Centric Cash smart contracts. The audit was performed between 03 April 2020 and 06 April 2020.

This report is organized into the following sections.

- **Section 2 - Executive Summary:** A high-level description of the findings of the audit.
- **Section 3 - Audit Details:** A description of the scope and methodology of the audit.
- **Section 4 - Design Specification:** An outline of the intended functionality of the smart contracts.
- **Section 5 - Detailed Findings:** Detailed descriptions of the findings of the audit.

The information in this report should be used to better understand the risk exposure of the smart contracts, and as a guide to improving the security posture of the smart contracts by remediating issues identified. The results of this audit are only a reflection of the source code reviewed at the time of the audit and of the source code that was determined to be in-scope.

The purpose of this audit was to achieve the following:

- Identify potential security flaws.
- Ensure that the smart contracts functioned according to the documentation provided.

Assessing the economics, game theory, or underlying business model of the platform were beyond the scope of this audit. Therefore, determining the viability of the deflationary currency or the effectiveness of the stabilizing mechanisms were beyond the scope of the audit.

Due to the unregulated nature and ease of transfer of cryptocurrencies, operations that store or interact with these assets are considered very high risk with regards to cyber attacks. As such, the highest level of security should be observed when interacting with these assets. This requires a forward-thinking approach, which takes into account the new and experimental nature of blockchain technologies. Strategies that should be used to encourage secure code development include:

- Security should be integrated into the development lifecycle and the level of perceived security should not be limited to a single code audit.
- Defensive programming should be employed to account for unforeseen circumstances.
- Current best practices should be followed where possible.

2. Executive Summary

This report presents the findings of an audit performed by iosiro on the Centric Rise and Centric Cash smart contracts. The Centric Foundation provided the Centric Whitepaper (MD5=a8b3c2e33ef3e36bcc1667cbdd0190a5) as a reference for the audit.

2.1 Important Information for Users of the Centric System

Users of the Centric system should be explicitly informed about the following characteristics of the system. - Admins are required to create Price Blocks. If an admin fails to create a Price Block for a point in time, the system will stop operating until a Price Block is created for that point in time. - The only exchange rate governed by the system is the Centric Rise to Centric Cash exchange rate. This rate is programmatically increased based on a Growth Rate set by the admin each block. At a smart contract level, the exchange rate is in no way related to a USD exchange rate. - The Price Factors that are used to establish the hourly increase in exchange rate, based on the Growth Rate, are calculated off-chain. This avoids unnecessarily implementing complex mathematical functionality on-chain; however, it means that there is the potential for Growth Rates and Price Factors to deviate from the algorithm defined in the whitepaper. As Growth Rates need to be locked before the first Price Block can be created, and cannot be modified after that point, users are recommended to verify that the Price Factors committed on-chain correspond to the Growth Rates according to the proposed algorithm.

2.2 Audit Results

At the conclusion of the audit, only informational level issues were open. These findings included security and functional recommendations that could be used to improve the system, either by enhancing the security model of the system or better adhering to best practices.

At a high level, the security posture of the smart contracts could be further strengthened by:

- Remediating the issues identified in this report and performing a review to ensure that the issues were correctly addressed.
- Performing additional audits at regular intervals, as security best practices, tools, and knowledge change over time. Additional audits over the course of the project's lifespan ensure the longevity of the codebase.
- Extending Centric's existing bug bounty program to encourage the responsible disclosure of security vulnerabilities in the smart contracts.

2.3 Audit Review

A review was performed on the 14th of April 2020 to verify changes made to the contracts based on the initial audit results, with the last commit at the time being 22a38fb. The majority of informational issues opened were found to be addressed.

3. Audit Details

3.1 Scope

The source code considered in-scope for the assessment is described below. Code from all other files is considered to be out-of-scope. Out-of-scope code that interacts with in-scope code is assumed to function as intended and introduce no functional or security vulnerabilities for the purposes of this audit.

3.1.1 Centric Smart Contracts

Project Name: contracts
Commits: 4a86292
Files: `Cash.sol`, `DateLib.sol`, `Rise.sol`, `RoundMath.sol`, `SafeMath.sol`, `TRC20.sol`, `helpers/Administrable.sol`, `helpers/Claimable.sol`

3.2 Methodology

A variety of techniques, described below, were used to conduct the audit.

3.2.1 Code Review

The source code was manually inspected to identify potential security flaws. Code review is a useful approach for detecting security flaws, discrepancies between the specification and implementation, design improvements, and high risk areas of the system.

3.2.2 Dynamic Analysis

The contracts were compiled, deployed, and tested in a Ganache test environment, both manually and through the Truffle test suite provided. Manual analysis was used to confirm that the code operated at a functional level and to verify the exploitability of any potential security issues identified. The coverage report of the provided Truffle tests is given below.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
Cash.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
DateLib.sol	100	100	100	100	
Rise.sol	100	100	100	100	
RoundMath.sol	100	100	100	100	
SafeMath.sol	100	100	100	100	
TRC20.sol	100	100	100	100	
contracts/helpers/	100	100	100	100	
Administrable.sol	100	100	100	100	
Claimable.sol	100	100	100	100	
contracts/mocks/	100	100	100	100	
CashBurnFromRiseMock.sol	100	100	100	100	
DateLibMock.sol	100	100	100	100	
RiseMock.sol	100	100	100	100	
RiseTransferMock.sol	100	100	100	100	
RoundMathMock.sol	100	100	100	100	
SafeMathMock.sol	100	100	100	100	
TRC20Mock.sol	100	100	100	100	
All files	100	100	100	100	

3.2.3 Automated Analysis

Tools were used to automatically detect the presence of several types of security vulnerabilities, including reentrancy, timestamp dependency bugs, and transaction-ordering dependency bugs. The static analysis results were manually analyzed to remove false-positive results. True positive results would be indicated in this report.

Static analysis was conducted using Slither, Securify, as well as MythX. Tools such as the Remix IDE, compilation output, and linters were also used to identify potential areas of concern.

3.3 Risk Ratings

Each issue identified during the audit has been assigned a risk rating. The rating is determined based on the criteria outlined below.

- **High Risk** - The issue could result in a loss of funds for the contract owner or system users.
- **Medium Risk** - The issue resulted in the code specification being implemented incorrectly.
- **Low Risk** - A best practice or design issue that could affect the security of the contract.
- **Informational** - A lapse in best practice or a suboptimal design pattern that has a minimal risk of affecting the security of the contract.
- **Closed** - The issue was identified during the audit and has since been addressed to a satisfactory level to remove the risk that it posed.

4. Design Specification

The following section outlines the intended functionality of the system at a high level. The specification is based on the implementation in the codebase and any perceived points of conflict should be highlighted with the auditing team to determine the source of the discrepancy.

4.1 Centric Rise

Centric Rise is a cryptocurrency with a deflationary supply.

TRC20 Token

Centric Rise should be represented by a TRC20-compliant token with the following values.

Field	Value
Name	Centric RISE
Symbol	CNR
Decimals	8

Minting

A total of 1 billion tokens should be minted to an address controlled by the Centric Foundation during the creation of Centric Rise.

Exchanging

By using the functions in the Centric Rise contract, it should be possible to exchange Centric Rise for Centric Cash and vice versa, at an exchange rate determined by the Centric Rise contract. When a user exchanges Centric Rise for Centric Cash, an equivalent amount of Centric Rise is *quarantined* in the Rise contract, according to a predetermined exchange rate set by the contract admins. When a user exchanges Centric Cash for Centric Rise, the Centric Cash amount is burned, and an equivalent amount of Centric Rise is sent from the quarantined amount in the Rise contract. As the algorithm should only ever output positive growth rates, there should not be any liquidity issues.

Exchange Rate

The exchange rate of Centric Rise to Centric Cash should be stored in Price Blocks. The exchange rate should be calculated based on the following algorithm.

```
vn = vl * (1+r)^(1/t)
```

- *vn* is the CNR price of the new Price Block.
- *vl* is the CNR price of the previous Price Block.
- *r* is the Growth Rate.
- *t* is the number of hours in the month the Price Block references.

Price Blocks

Price Blocks should store the current, future, and past exchange rates of Centric Rise and Centric Cash. It should only be possible for an admin (i.e. Centric team) to create Price Blocks, and only once the Growth Factors have been locked. The admin will need to specify the Block Number and the Growth Rate when creating the Price Block. Once created, it should not be possible to modify or remove the Price Block. Each Price Block is referenced by a Block Number that corresponds to an hour in time, denoted in hours since Unix epoch. It should be possible to create any number of Price Blocks ahead of time. Price Blocks should be readable by any user. Each Price Block should contain: * CNR price denominated in CNS. * Monthly price growth rate,

expressed as a percentage. * CNR price change from the previous Price Block, expressed as a percentage. * Created hour denoted in hours since Unix epoch.

Growth Rate

The Growth Rate is the rate at which the exchange rate of Centric Rise to Centric Cash increases over a one month period. Admins can call the `setPriceFactors(uint256 _growthRate, uint256[4] _priceFactors)` function to set the growth rate and price factors. Growth Rates and Price Factors need to be set before Price Blocks can be created, and can not be modified after that time. The Growth Rate should be more than 0% and less than 100%.

Price Factors

The Price Factors are passed into `setPriceFactors(uint256 _growthRate, uint256[4] _priceFactors)` function by an admin and rudimentary validation is performed in order to check that the values are within certain bounds to match the validation performed on the Growth Rate. The Price Factors are not calculated on-chain, but should be calculated by admins as:

```
_priceFactor[0] = (1+_growthRate/GROWTH_RATE_BASE)^(1/(28*24))
_priceFactor[1] = (1+_growthRate/GROWTH_RATE_BASE)^(1/(29*24))
_priceFactor[2] = (1+_growthRate/GROWTH_RATE_BASE)^(1/(30*24))
_priceFactor[3] = (1+_growthRate/GROWTH_RATE_BASE)^(1/(31*24))
```

Balancing

It should be possible for any account to burn Centric Rise held by the Rise contract, in excess of the required quarantined amount of Centric Rise. This should allow for the deflation of the currency to be tracked through the total supply.

Burn Lost Tokens

The contract owner should be able to burn any Centric Rise held by the Rise smart contract that is in excess of the required quarantined amount of Centric Rise.

4.2 Centric Cash

Centric Cash is intended to be traded on exchanges at market rates. It can be converted to Centric Rise and vice versa.

TRC20 Token

It should be represented by a TRC20 compliant token with the following values.

Field	Value
Name	Centric CASH
Symbol	CNS
Decimals	8

Minting

Centric Cash tokens should only be minted by the Rise contract when a user quarantines Centric Rise. The amount of Centric Cash issued should be based on the Centric Rise to Centric Cash exchange rate in the Rise contract.

Burning

When exchanging Centric Cash for Centric Rise, the Centric Cash amount should be burned from the total supply.

5. Detailed Findings

The following section details the findings of the audit.

5.1 High Risk

No high risk issues were present at the conclusion of the review.

5.2 Medium Risk

No medium risk issues were present at the conclusion of the review.

5.3 Low Risk

No low risk issues were present at the conclusion of the review.

5.4 Informational

5.4.3 Design Comments

Actions to improve the functionality and readability of the codebase are outlined below.

Use CNS instead of USD in Comments and Documentation

In the comments of Rise.sol, several references to the Centric Rise price being denominated in USD were made. For the sake of clarity and transparency, it is advised that these USD references be changed to CNS. No input of USD was made into the smart contract system, and the only exchange rate present in the smart contracts was between CNR and CNS. As this rate was calculated based on an algorithm that produced only increasing values, it could not appropriately model the real CNR:USD or CNS:USD exchange rate.

Secure Sensitive Functionality

Generally, the Centric system managed to minimize the use of trusted roles, and included assertions to minimize the potential for invalid or malicious values to be used.

However, the contract owner and admin roles still had the potential to disrupt the functioning of the system in the event of compromised or lost keys. This was a centralized, single point of failure risk. For example, admins were required to initiate the creation of Price Blocks. Losing this ability would mean a system-wide denial of service. The Centric Foundation indicated that their strategy would be to create a year's worth of Price Blocks at a time, in order to minimize the risk of accidentally reaching a state without a valid Price Block. However, this case could still be encountered.

The risk of a single rogue actor or a compromised private key of an owner or admin account can be mitigated by ensuring that a high standard of security is observed by sensitive accounts. For example, they should be multi-signature and signed by properly secured hardware-backed accounts.

5.5 Closed

5.5.1 Insufficient Unit Test Quality and Coverage (Informational)

General

Certain functionality was found to have insufficient unit test quality and coverage. To improve the maintainability of the code, and decrease the likelihood of introducing functional or security issues into the codebase, the test suite should be extended to cover the following:

- Rise.sol#L372-379,L403-404

- DateLib.sol#L75-80,L92-95

Review Update

Fixed in 22a38fb. The test suite was extended to cover the previously uncovered functionality.

5.5.2 Design Comments (Informational)

Actions to improve the functionality and readability of the codebase are outlined below.

Validate Starting Price Block Time

When first calling the `doCreateBlock(uint256 _blockNumber, uint256 _growthRate)` function, the caller was required to pass a value in for `_blockNumber` that was greater than the current hour. This meant that only values in the future could be specified. However, there was no upper bound for the hour that could be specified, unnecessarily opening up the potential to accidentally set an incorrect value for the starting block.

It is recommended that validation be added to ensure that the `_blockNumber` is some reasonable amount of time in the future.

Review Update

Recommendation applied in c72f535. `_blockNumber` is required to be within one year ahead of the current hour.

Code Style Improvements

The following changes are recommended to improve the readability of the codebase. - Rise.sol#L35: `initialPrice` should be `INITIAL_PRICE` as it is a constant variable. - Rise.sol#L330: `burnQuarantined()` should be `_burnQuarantined()` as it is an internal function. - Rise.sol#L350: `createBlock(...)` should be `_createBlock(...)` as it is an internal function.

Review Update

Suggested changes applied in ae3d93e.

Centric Cash Smart Contract Audit

Centric Foundation, 16 April 2020

1. Introduction

iosiro was commissioned by the Centric Foundation to conduct a smart contract audit on the Centric Cash smart contract. The audit was performed on 16 April 2020.

This report is organized into the following sections.

- **Section 2 - Executive Summary:** A high-level description of the findings of the audit.
- **Section 3 - Audit Details:** A description of the scope and methodology of the audit.
- **Section 4 - Design Specification:** An outline of the intended functionality of the smart contracts.
- **Section 5 - Detailed Findings:** Detailed descriptions of the findings of the audit.

The information in this report should be used to better understand the risk exposure of the smart contracts, and as a guide to improving the security posture of the smart contracts by remediating issues identified. The results of this audit are only a reflection of the source code reviewed at the time of the audit and of the source code that was determined to be in-scope.

The purpose of this audit was to achieve the following.

- Identify potential security flaws.
- Ensure that the smart contracts functioned according to the documentation provided.

Assessing the economics, game theory, or underlying business model of the platform were beyond the scope of this audit. Therefore, determining the viability of the deflationary currency or the effectiveness of the stabilizing mechanisms to hold the 1 USD peg were explicitly beyond the scope of the audit.

Due to the unregulated nature and ease of transfer of cryptocurrencies, operations that store or interact with these assets are considered very high risk with regards to cyber attacks. As such, the highest level of security should be observed when interacting with these assets. This requires a forward-thinking approach, which takes into account the new and experimental nature of blockchain technologies. Strategies that should be used to encourage secure code development include:

- Security should be integrated into the development lifecycle and the level of perceived security should not be limited to a single code audit.
- Defensive programming should be employed to account for unforeseen circumstances.
- Current best practices should be followed where possible.

2. Executive Summary

This report presents the findings of an audit performed by iosiro on the Centric Cash smart contracts. The scope of the audit was limited to only the basic TRC20 functionality.

It should be noted that the Centric Rise contract, which was entirely responsible for controlling the supply (i.e. minting and burning Centric Cash), was audited separately. The audit of the Centric Rise system performed by iosiro is publicly available here TODO: UPDATE ONCE LIVE.

The Centric Foundation provided the Centric Whitepaper (MD5=a8b3c2e33ef3e36bcc1667cbdd0190a5) as a reference for the audit.

No issues were open at the conclusion of the audit.

At a high level, the security posture of the smart contracts could be further strengthened by:

- Performing additional audits at regular intervals, as security best practices, tools, and knowledge change over time. Additional audits over the course of the project's lifespan ensure the longevity of the codebase.
- Extending Centric's existing bug bounty program to encourage the responsible disclosure of security vulnerabilities in the smart contracts.

3. Audit Details

3.1 Scope

The source code considered in-scope for the assessment is described below. Code from all other files is considered to be out-of-scope. Out-of-scope code that interacts with in-scope code is assumed to function as intended and introduce no functional or security vulnerabilities for the purposes of this audit.

3.1.1 Centric Smart Contracts

Project Name: contracts
Commits: 22a38fb
Files: Cash.sol, SafeMath.sol, RoundMath.sol, TRC20.sol, helpers/Administrable.sol, helpers/Claimable.sol

3.2 Methodology

A variety of techniques were used while conducting the audit. These techniques are briefly described below.

3.2.1 Code Review

The source code was manually inspected to identify potential security flaws. Code review is a useful approach for detecting security flaws, discrepancies between the specification and implementation, design improvements, and high risk areas of the system.

3.2.2 Dynamic Analysis

The contracts were compiled, deployed, and tested in a Ganache test environment, both manually and through the Truffle test suite provided. Manual analysis was used to confirm that the code operated at a functional level, and to verify the exploitability of any potential security issues identified. The coverage report generated by solidity-coverage of the Truffle tests included in the project is given below.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
Cash.sol	100	100	100	100	
RoundMath.sol	100	100	100	100	
SafeMath.sol	100	100	100	100	
TRC20.sol	100	100	100	100	
contracts/helpers/	100	100	100	100	
Administrable.sol	100	100	100	100	
Claimable.sol	100	100	100	100	
All files	100	100	100	100	

3.2.3 Automated Analysis

Tools were used to automatically detect the presence of several types of security vulnerabilities, including reentrancy, timestamp dependency bugs, and transaction-ordering dependency bugs. The static analysis results were manually analyzed to remove

false-positive results. True positive results would be indicated in this report. Static analysis was conducted using Slither, Securify, as well as MythX. Tools such as the Remix IDE, compilation output, and linters were also used to identify potential areas of concern.

3.3 Risk Ratings

Each issue identified during the audit has been assigned a risk rating. The rating is determined based on the criteria outlined below.

- **High Risk** - The issue could result in a loss of funds for the contract owner or system users.
- **Medium Risk** - The issue resulted in the code specification being implemented incorrectly.
- **Low Risk** - A best practice or design issue that could affect the security of the contract.
- **Informational** - A lapse in best practice or a suboptimal design pattern that has a minimal risk of affecting the security of the contract.
- **Closed** - The issue was identified during the audit and has since been addressed to a satisfactory level to remove the risk that it posed.

4. Design Specification

The following section outlines the intended functionality of the system at a high level. The specification is based on the implementation in the codebase and any perceived points of conflict should be highlighted with the auditing team to determine the source of the discrepancy.

4.1 Centric Cash

Centric Cash is a cryptocurrency intended to be traded on exchanges at market rates.

TRC20 Token

It should be represented by a TRC20 compliant token with the following values.

Field	Value
Name	Centric CASH
Symbol	CNS
Decimals	8

Minting

It should only be possible for the Centric Rise contract to mint Centric Cash tokens. The Centric Rise contract should be able to freely mint an arbitrary amount of Centric Cash tokens to an address.

Burning

It should only be possible for the Centric Rise contract to burn Centric Cash tokens. The Centric Rise contract should be able to burn an arbitrary amount of Centric Cash tokens from any address. When burning Centric Cash tokens, the amount should be removed from the total supply.

Centric Rise Contract

It should be possible for only the Centric Cash contract owner to set the address of the Centric Rise contract. It should not be possible to change the address once set.

5. Detailed Findings

The following section details the findings of the audit.

5.1 High Risk

No high risk issues were present at the conclusion of the review.

5.2 Medium Risk

No medium risk issues were present at the conclusion of the review.

5.3 Low Risk

No low risk issues were present at the conclusion of the review.

5.4 Informational

No informational risk issues were present at the conclusion of the review.