Synthetix Sargas Release Smart Contract Audit

1. Introduction

iosiro was commissioned by Synthetix to conduct a smart contract audit of their Sargas Release, which included audits on the following components:

- SIP-135 from 2 September until 10 September, and a review on 27 and 30 September, consuming a total of 9 resource days.

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 understand the smart contracts' risk exposure better and as a guide to improving the security posture of the smart contracts by remediating issues identified. The results of this audit reflect the in-scope source code reviewed at the time of the audit.

The purpose of this audit was to achieve the following:

- Ensure that the smart contracts functioned as intended.
- Identify potential security flaws.

Assessing the off-chain functionality associated with the contracts, for example, backend web application code, was outside of the scope of this audit.

Due to the unregulated nature and ease of transfer of cryptocurrencies, operations that store or interact with these assets are considered high risk from 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.
- Defensive programming should be employed to account for unforeseen circumstances.
- Current best practices should be followed when possible.

2. Executive summary

This report presents the findings of the audit performed by iosiro of the smart contract implementation of the Sargas release.

SIP-135 refactored the existing multi-collateral loans and shorts to be compatible with Optimism and added some additional functionality. Two low and several

informational issues were identified and addressed during the audit. The issues pertained to optimizations, code consistency, and adhering to best practices.

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 Synthetix SIP-135 smart contracts

Project name: Synthetix
Commits: fa160ff, c05d20e, bbbb0c3, 645a49c
Files: contracts/Collateral.sol, contracts/CollateralErc20.sol, contracts/CollateralManager.sol, contracts/CollateralShort.sol, contracts/CollateralState.sol, contracts/MixinSystemSettings.sol, contracts/MixinSystemSettings.sol, contracts/SystemSettings.sol

3.2 Methodology

The audit was conducted using a variety of techniques 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 test environment, both manually and through the test suite provided. Manual analysis was used to confirm that the code was functional and discover security issues that could be exploited.

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. Static analysis results were reviewed manually and any false positives were removed. Any true positive results are included in this report.

Static analysis tools commonly used include Slither, Securify, and MythX. Tools such as the Remix IDE, compilation output, and linters could also be 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 satisfactorily addressed, removing the risk it posed.

4. Design specification

The following section outlines the intended functionality of the system at a high level.

4.1 SIP-135

The specification of SIP-135 was based on commit hash 4878372.

5. Detailed findings

The following section includes in-depth descriptions of 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 identified low-risk issues were open at the conclusion of the review.

5.4 Informational

No identified informational issues were open at the conclusion of the review.

5.5 Closed

5.5.1 Not using safe ERC-20 transfers (low risk)

CollateralErc20.sol

Description

When performing an ERC-20 transfer, the return value of the transfer should be validated to ensure that the transfer succeeded. The CollateralErc20 functions `open`, `close`, `deposit`, `withdraw`, and `liquidate` functions did not correctly validate the return value, and as such, may result in erroneous behavior in the system.

Recommendation

The OpenZeppelin SafeERC20 library should be used to wrap all calls to `transfer` and `transferFrom`.

Update

Implemented in c05d20e.

5.5.2 Implement skew limit (low risk)

CollateralManager.sol#L241

Description

The `getShortRate` used the ratio of shorts to longs for a synth to determine the current short rate for the synth. In the event that shorts far exceeded longs, the rate would tend towards the value of the shorts issued, which appeared to be excessive.

Recommendation

A limit should be considered to cap the maximum short rate possible.

Update

A skew factor was implemented in a4d1631 to scale the short rate. It should be noted that the short rate could still become fairly large in the scenario outlined above.

5.5.3 Design comments (informational)

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

Interest accrual only approximates real-world values

As highlighted in a previous audit report, the `accrueInterest` function is only an approximation of the real rate of interest accrual. The more frequently the function is called, the more accurate the approximation will be. As this implementation is intended to be released on Optimism, where the rate of interaction should presumably be higher, the issue should be less relevant than before and has been closed, but is still raised for completeness.

Remove unused contract

The `CollateralState.sol` file is no longer used and should be removed.

Update

Removed in c05d20e.

Gas optimizations

Collateral.sol#L427

The `_liquidate` function does not need to explicitly call `_isLoanOpen` as the `_getLoanAndAccrueInterest` function will also call it.

Update

Removed in c05d20e.

Missing service fee

The service fee described in the specification was not implemented in the code.

Update

The service fee was implemented in c05d20e.

Improve comments

The following comments could be improved to describe the code better.

- Collateral.sol#L559 "let the child handle it" is incorrect as the child contract is no longer called.
- Collateral.sol#L588 "Otherwise multiple" → multiply.
- CollateralManager.sol#L102 "iSynth" should be "ISynth" to avoid confusion with inverse synths.

Update

Comments were amended in c05d20e.

Potential gas issues

CollateralManager.sol#L274

As the `exceedsDebtLimit` calculation needs to make external calls to each of the long and short synths, this function will likely run into the gas limit on Optimism.

Update

Function reworked in 645a49c.

Max decimal assumption

There is an assumption in the ERC-20 implementation that token collaterals will have a maximum of 18 decimals. Tokens with more than 18 decimals will revert when trying to open loans.

Update

Acknowledged by the Synthetix team.

Gas optimization

CollateralShort.sol#L61

The `closeWithCollateral` function can include a check for whether `collateral > 0` to avoid attempting 0 value transfers.

Update

Implemented in c05d20e.

Missing revert message

CollateralManager.sol#L489

The revert performed in the `accrueInterest` function is missing a revert message. A revert message should be included to help inform users of the reason for the revert.

Update

Implemented in c05d20e.

Missing event

CollateralManager.sol#L289

An event should be emitted when the `setUtilisationMultiplier` function is called.

Update

Implemented in c05d20e.