Synthetix Kaus Release Smart Contract Audit

1. Introduction

iosiro was commissioned by Synthetix to conduct a smart contract audit of their Kaus Release, which included an audit of the following component:

- SIP-187 on 11 October, consuming 1 resource day.

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 risk exposure of the smart contracts, and as a guide to improving the security posture of the smart contracts by remediating the issues that were 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:

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

Assessing the market effect, economics, game theory, or underlying business model of the platform were strictly beyond 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 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. There are a number of techniques that can help to achieve this, some of which are described below.

- 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 smart contract audits performed by iosiro of Synthetix's Kaus release.

SIP-187 implemented a fix for an issue where the debt cache was not properly updated through partial debt updates when burning or issuing sUSD. One informational issue was identified and remediated during the audit.

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-187 Smart Contracts

Project Name: Synthetix

Commits: 0bae287, 8dbad19, 81d0d4f

Files: contracts/DebtCache.sol, contracts/Issuer.sol, contracts/SafeCast.sol, contracts/SafeDecimalMath.sol

3.2 Methodology

A variety of techniques were used in order to perform 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 test environment, both manually and through the 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.

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 tools commonly used include Slither, Securify, and MythX. Tools such as the Remix IDE, compilation output, and linters are also used to identify potential issues.

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.

4.1 SIP-187

The specification of SIP-187 was based on commit hash 44d02fb.

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 audit.

5.2 Medium Risk

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

5.3 Low Risk

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

5.4 Informational

No informational issues were present at the conclusion of the audit.

5.5 Closed

5.5.1 Potential overflow when casting between int to uint (Low Risk)

Issuer.sol#L671

Issuer.sol#L697

Care should be taken when casting between `int256` and `uint256` to ensure that the value being cast does not result in potential overflows. It is recommended to manually validate that the values are within bounds or use the OpenZeppelin SafeCast functionality.

Update

SafeCast was used in 8dbad19.

Secure your system.

Request a service

START NOW →