Synthetix Aloith Release Smart Contract Audit

1. Introduction

iosiro was commissioned by Synthetix to conduct a smart contract audit of the implementation of SIP-112 and SIP-136 for the Aloith Release. The audit was performed as follows:

- SIP-112: two auditors on 21 to 27 April consuming a total of ten resource days.
- SIP-136: one auditor on 10 to 13 May consuming a total of four 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 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 the audit performed by iosiro on the implementation of SIP-112 and SIP-136.

SIP-112 introduced the ability for users to move between sETH and ETH or WETH, with a fee taken for the fee pool. Two low risk issues were identified and remediated during the audit.

SIP-136 made improvements to the debt pool and debt cache calculations. One low risk 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-112 smart contracts

Project Name: Synthetix
Commits: b16096e
Files: contracts/BaseDebtCache.sol, contracts/BaseSynthetix.sol, contracts/EtherWrapper.sol, contracts/FeePool.sol, contracts/MixinSystemSettings.sol, contracts/MultiCollateralSynth.sol, contracts/NativeEtherWrapper.sol, contracts/SystemSettings.sol,

3.1.2 Synthetix SIP-136 smart contracts

Project Name: Synthetix
Commits: 1159c8a
Files: contracts/BaseDebtCache.sol, contracts/DebtCache.sol, contracts/Issuer.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 Ganache 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-112

The specification of SIP-112 was based on commit hash 668e839.

4.1 SIP-136

The specification of SIP-136 was based on commit hash 668e839.

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 Incorrect invalid rate logic (low risk)

SIP-116: BaseDebtCache.sol#L154, BaseDebtCache.sol#L200, BaseDebtCache.sol#L218, DebtCache.sol#L96

Description

During debt calculations, when determining whether any rates were invalid, the logical `AND` operator was used on different invalid rates, which would determine whether both variables were invalid. However, in order to correctly determine whether any rates were invalid the `OR` operator should have been used.

Recommendation

The invalid should be determined by using `OR` instead of `AND`.

Update

Implemented in 480f395.

5.5.2 sETH rate may be invalid when remitting fee (low risk)

*SIP-112: EtherWrapper.sol#L228

Description

When remitting the fee, the `exchangeRates().effectiveValue(...)` function is used to determine the USD amount of the sETH being remitted. However, the sETH rate is not validated to ensure that it is valid. As a result, the incorrect amount of fees may be minted.

Recommendation

The current sETH rate should be validated to ensure that it is valid.

```
require(!exchangeRates().rateIsInvalid(ETH))
```

Update

Implemented in fbacc27.

5.5.3 Use of `transfer` function (low risk)

SIP-112: NativeEtherWrapper.sol#L81

Description

The `burn()` function made use of the `transfer()` function to send ether. While `transfer()` is commonly used to prevent reentrancy attacks due to its 2300 gas limit, it relies on the receiving contract to have a fallback function below this limit. As demonstrated in EIP-1884, which changed the gas cost of the `SLOAD` operation, gas costs can change. This could lead to a case where a contract has its fallback function increased above the 2300 limit, resulting in it becoming incompatible with the system. More information can be found in Consensys On Avoiding `transfer()`.

Recommendation

It is recommended that the `call()` function be used to send ether instead of `transfer()`.

Update

Implemented in d8af716.

Secure your system.

Request a service

START NOW →

ABOUT

CONTACT US

SMART CONTRACT AUDITING

PENETRATION TESTING

AUDIT REPORTS

PRIVACY POLICY

TERMS OF SERVICE