Synthetix Fomalhaut Release Smart Contract Audit

1. Introduction

iosiro was commissioned by Synthetix to conduct a smart contract audit on the Fomalhaut release, including SIP-77, SIP-85, and SIP-86. The audit was performed between 18 September 2020 and 23 September 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 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 smart contract implementation of the Synthetix Fomalhaut Release.

SIP-77

SIP-77 applied bug fixes to the `StakingRewards` contract and added the ability to pause staking.

The implementation accorded with the specification provided and no issues were identified with the bug fix.

SIP-85

SIP-85 implemented the third iteration of the Ether collateral trial. This iteration introduced borrowing and issuing sUSD against Ether collateral.

Several issues were identified and closed during the audit. One informational issue remained open at the conclusion of the audit. Overall, at the conclusion of the audit the implementation was of a high standard and accorded with the specification provided.

SIP-86

The purpose of SIP-86 was to update the `ExchangeRates` contract to use the ChainLink Aggregator V2V3 Interface. The hybrid interface includes functions from both V2 and V3 interfaces, allowing the `ExchangeRates` contract to make use of new functionality, while maintaining compatibility with the old interface where needed. Furthermore,

as the ChainLink aggregator view functions could revert if passed an unknown `roundID`, a low level `staticcall` was implemented to prevent Synthetix functions from reverting when interacting with it.

One informational issue was identified and closed by the conclusion of the audit. Overall, the implementation was of a high standard.

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-77 Smart Contracts

Project Name: Synthetix
Commits: bb4c0f3
Files: StakingRewards.sol

3.1.2 Synthetix SIP-85 Smart Contracts

Project Name: Synthetix
Commits: b1bbcf8
Files: EtherCollateralsUSD.sol, FeePool.sol, Issuer.sol

3.1.2 Synthetix SIP-86 Smart Contracts

Project Name: Synthetix
Commits: 95d73f7
Files: ExchangeRates.sol

Project Name: Synthetix
Commits: 09a0e01
Files: ExchangeRates.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-77

The specification of SIP-77 was based on commit hash f15c658.

4.2 SIP-85

The specification of SIP-85 was based on commit hash da5fe13.

4.3 SIP-86

The specification of SIP-86 was based on commit hash f15c658.

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

5.4.1 Use of `transfer` function

SIP-85: EtherCollateralsUSD.sol#L504, EtherCollateralsUSD.sol#L588, EtherCollateralsUSD.sol#L665, EtherCollateralsUSD.sol#L669

Description

The `withdrawCollateral`, `liquidateLoan` and `_closeLoan` functions 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.

Remedial Action

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

Further Reading

Consensys On Avoiding `transfer()`.

5.5 Closed

5.5.1 Loan Debt Cleared Without Full Repayment (High Risk)

SIP-85: EtherCollateralsUSD.sol#L665

Description

A user could clear their loan debt without full repayment by paying an amount smaller than the loan's accrued interest.

This vulnerability relied on the `loanAmountAfter` not being set when calling the `_splitInterestsAndLoanPayment`. By paying an amount smaller than `_accruedInterest`, the `remainingPayment` amount would be zero and not trigger the following if-statement. This caused the `loanAmountAfter` to be returned with the default value of 0. The `loanAmountAfter` was then set as the current debt balance with a call to `_updateLoan`.

Remedial Action

It is recommended that the `remainingPayment > 0` conditional statement be removed and perform the loan payment regardless of the value of `remainingPayment`. Alternatively, the `loanAmountAfter` should be initialized with the `_loanAmount` before the loan payment.

Update

Implemented in 22fe386.

5.5.2 Liquidated Loan Collateral Not Updated (High Risk)

Description

SIP-85: EtherCollateralsUSD.sol#L580

When a loan was liquidated with the `liquidateLoan` function, the underlying loan collateral was not updated to reflect the liquidation.

When a loan is partially liquidated, the liquidator receives the loan's underlying collateral with an additional reward in the form of a liquidation penalty. Both the liquidated collateral and the liquidation penalty were not updated in the user's loan, which allowed the user to lend further capital up to the remaining collateral amount. Furthermore, a user could liquidate their own loans to exhaust the funds from the underlying collateral pool.

Remedial Action

It is recommended to update the loan's collateral value to reflect the new amount after liquidation.

Update

Implemented in f0bfe83.

5.5.3 Incorrect `amountToLiquidate` Calculation (Medium Risk)

Description

SIP-85: EtherCollateralsUSD.sol#L568

When liquidating loans at or below the issuance ratio, the contract may burn synths more than the intended amount of `_debtToCover`.

The `amountToLiquidate` calculation returned the `liquidationAmount` if it was more than the `_debtToCover` value. The `amountToLiquidate` was then used as the amount of sUSD to burn from the user, which was more than the `_debtToCover` the user intended to burn to liquidate the loan.

Remedial Action

It is recommended to limit the `_debtToCover` to be at most the `liquidationAmount`.

Update

Implemented in f0bfe83.

5.5.4 Unsafe Arithmetic Used (Informational)

SIP-86: ExchangeRates.sol#L548, ExchangeRates.sol#L549

Description

The `_formatAggregatorAnswer` function made use of `*` and `-` operators instead of `SafeMath.mul()` and `SafeMath.sub()` respectively. It was unlikely that the use of `*` or `-` would cause an overflow or underflow in `_formatAggregatorAnswer` as the `currencyKeyDecimals[currencyKey]` amount was validated in the `addAggregator` function.

Remedial Action

It is recommended to make use of `SafeMath.mul()` in place of `*` and `SafeMath.sub()` in place of `-`.

Update

Implemented in 749d1c6.

5.5.5 Design Comments (Informational)

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

Fix Spelling and Grammatical Errors

Language mistakes were identified in the codebase. Fixing these mistakes can help improve the end-user experience by providing clear information on errors encountered, and improve the maintainability and auditability of the codebase.

SIP-86

1. EtherCollateralsUSD.sol#L55: `Min 1ETH =` should be `Min 1 ETH`.
2. EtherCollateralsUSD.sol#L394: `limted` should be `limited`.
3. EtherCollateralsUSD.sol#L654: `liquidatior` should be `liquidator`.
4. EtherCollateralsUSD.sol#L654: `redeeemed` should be `redeemed`.

Update

Fixed in d7cb8d1.

Refactoring Suggestions

It is recommended that certain portions of the code be refactored to improve readability and consistency, as indicated below.

SIP-85

- In Solidity version 0.7.0, the `now` keyword was deprecated. Developers are encouraged to use `block.timestamp` instead, to ensure forward compatibility.
- The `HARD_CAP` should be declared as a constant state variable.

Update

Implemented in 48623b5 and d7cb8d1.

Incorrect Function Description

SIP-85: EtherCollateralsUSD.sol#L472

The `withdrawCollateral` function had the description `Add ETH collateral to an open loan` when it should be `Withdraw ETH collateral from an open loan.`

Update

Fixed in d7cb8d1.

Code Reuse

SIP-85 EtherCollateralsUSD.sol#L460

The `depositCollateral` function did not reuse available code when checking if a `synthLoan` exists and is open. The validation should be done by calling the `_checkLoanIsOpen()` function.

Update

Implemented in f29ffde.