SFP Pilot Audit

Strands, 27 February 2024

Confidential
17 Tyrwhitt Avenue, Johannesburg, 2196, South Africa
hello@iosiro.com

Contents

1. Introduction

iosiro was commissioned by Lyra (https://www.lyra.finance/) to conduct a smart contract audit of Strands Finance's SFP Pilot contracts. The audit was performed by 2 auditors between 26 and 27 February and 2024, using 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 smart contracts' risk exposure better and as a guide to improving the security posture of the smart contracts by remediating the 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:

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

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, 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 Strands Finance's SFP Pilot contracts

Audit findings

iosiro made two informational findings regarding the protocol's centralization risk and an unused contract in the code repository.

Recommendations

At a high level, the security posture of Strands Finance's 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 throughout the project's lifespan ensure the longevity of the codebase.
- Creating a bug bounty program to encourage the responsible disclosure of security vulnerabilities in the system.

3. Audit details

3.1 Scope

The source code considered in-scope for the assessment is described below. Code from all other files was considered to be out-of-scope. Out-of-scope code that interacts with in-scope code was assumed to function as intended and not introduce any functional or security vulnerabilities for the purposes of this audit.

3.1.1 Smart contracts

- Project name: sfp-pilot
- Commit: fcaaaa1
- Files: StrandsAPI.sol, StrandsSFP.sol, DecimalMath.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. This specification is based on the implementation in the codebase. Any perceived points of conflict should be highlighted with the auditing team to determine the source of the discrepancy.

The SFP Pilot project consists of a standard ERC20 token and an ERC4646 vault. Both are implemented using the OpenZeppelin library's implementations of the standards with minimal customization.

The primary enhancement to the reference ERC20 implementation is the introduction of both an Admin and Minter role using OpenZeppelin's Role-Based Access Control contract.

- **Admin role:** Can assign the Minter role to any address
- **Minter role:** Can mint and burn tokens. Tokens can only be burned from the Minter's own balance and not on behalf of any other token holder.

The ERC4646 vault is standard without any privileged roles or functionality.

5. Detailed findings

The following section details the findings of the audit.

5.1 High-risk findings

No high-risk issues were discovered during the audit.

5.2 Medium-risk findings

No medium-risk issues were discovered during the audit.

5.3 Low-risk findings

No low-risk issues were discovered during the audit.

5.4 Informational findings

5.4.1 Centralization risk

General

Description

The supply of Strands API token is fully controlled by addresses with the token's `MINTER_ROLE`. Addresses with this role can mint arbitrary amounts of API token at any time and deposit these holdings into the `StrandsSFP` vault to receive SFP tokens.

For the SFP token to maintain its expected value of $1, additional controls beyond the scope of the contracts reviewed in the SFP Pilot codebase must be imposed by Strands Finance. These controls must ensure that minters can only mint API tokens equivalent to their deposited fiat amounts.

If any of the External Owned Accounts with the MINTER_ROLE were to be compromised, the total supply of SFP tokens and their underlying API tokens would devalue.

Recommendation

Mechanisms for ongoing verification and publication of the proof of reserves for Strands Finance tokens should be defined. Circle's regular USDC proof of reserves report could be taken as a model to emulate.

Additional controls to minimize the risk of centralized minters should be investigated. Possible on-chain solutions include:

- Requiring `MINTER_ROLE` to be associated with a multisig control by various independent parties.
- Verification of an attestation signed by an independent party for deposits in the Segregated Fund Account.
- Zero-knowledge proof of the Meta-Layer-Encoder computation.
- Verification of trusted execution of Meta-Layer-Encoder computation (SGX attestation, Wormhole VAA, etc.)

In the short-term, the risk can be reduced by implementing a limit on the total supply of tokens. Only the `defaultAdmin` should be allowed to increase the limit based on the funds available in the Segregated Fund Account.

5.4.2 Unused contract

DecimalMath.sol

Description

The DecimalMath library was included in the contract repository but not imported or used by any protocol contracts.

Recommendation

DecimalMath.sol can be safely removed from the repository.

5.5 Closed findings

No findings were closed at the conclusion of the assessment.