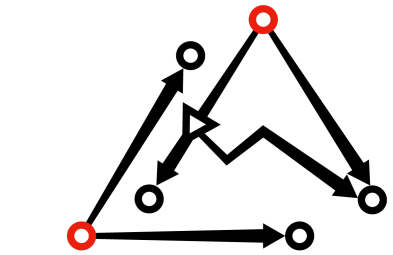

Least Authority

PRIVACY MATTERS

Metapool Smart Contracts

Security Audit Report

Thesis

Updated Final Audit Report: 30 November 2022

Table of Contents

Overview

Background

Thesis has requested that Least Authority perform a security audit of their Saddle Finance Metapool smart contracts.

Project Dates

- **July 4 - August 3:** Initial Code Review *(Completed)*
- **August 5:** Delivery of Initial Audit Report *(Completed)*
- **November 7 - November 23:** Verification Review *(Completed)*
- **November 29:** Delivery of Final Audit Report *(Completed)*
- **November 30:** Delivery of Updated Audit Report *(Completed)*

Review Team

- John Amatulli, Security Researcher and Engineer
- Nikos Iliakis, Security Researcher and Engineer
- Nishit Majithia, Security Researcher and Engineer
- Xenofon Mitakidis, Security Researcher and Engineer
- Akunne Obinna, Security Researcher and Engineer

Coverage

Target Code and Revision

For this audit, we performed research, investigation, and review of the Saddle Finance Metapool smart contracts followed by issue reporting, along with mitigation and remediation instructions as outlined in this report.

The following code repository is considered in-scope for the review:

- Metapool smart contracts: https://github.com/saddle-finance/saddle-contract/tree/master/contracts/meta

Specifically, we examined the Git revision for our initial review:

`49734a33ebd113b5befc824822f0f9d150ad2e76`

For the verification, we examined the Git revision:

`64d29f54e9397e4958b7546bc22bff4da754e24e`

For the review, this repository was cloned for use during the audit and for reference in this report:

- Metapool smart contracts: https://github.com/LeastAuthority/Metapool-Smart-Contracts/tree/master/contracts/meta

All file references in this document use Unix-style paths relative to the project's root directory.

In addition, any dependency and third-party code, unless specifically mentioned as in-scope, were considered out of scope for this review.

Supporting Documentation

The following documentation was available to the review team:

- About Saddle: https://docs.saddle.finance/

Areas of Concern

Our investigation focused on the following areas:

- Correctness of the implementation;
- Common and case-specific implementation errors;
- Adherence to the specification and best practices;
- Adversarial actions and other attacks on the smart contracts;
- Potential misuse and gaming of the smart contracts;
- Attacks that impact funds, such as the draining or manipulation of funds;
- Mismanagement of funds via transactions;
- Denial of Service (DoS) and other security exploits that would impact the intended use of the smart contacts or disrupt their execution;
- Vulnerabilities in the smart contract code;
- Protection against malicious attacks and other ways to exploit the smart contracts;
- Inappropriate permissions and excess authority;
- Data privacy, data leaking, and information integrity; and
- Anything else as identified during the initial analysis phase.

Findings

General Comments

Saddle Finance is a DeFi platform on the Ethereum blockchain that provides several services, including swaps, participation in liquidity pools, yield farming, and others. The Metapool smart contracts compose a component within the system's smart contract suite governing the functionality of Metapools, which pair a saddle stable coin with an underlying token, enabling swaps and pool participation rewards.

Our team reviewed the design and implementation of the Metapool smart contracts and did not identify any security vulnerabilities. Note that, Saddle Finance is composed of multiple components, which were out of scope and were assumed to function as intended for the purpose of this review.

System Design

We found that security has been taken into consideration in the design of the Metapool smart contracts as demonstrated by the utilization of necessary access controls and the use of modifiers to block potential attack vectors.

The Saddle Finance smart contracts are owned by an externally owned account (EOA) whereby the system's admin functionality is controlled by a Gnosis Safe Multi-Sig, which can call for the emergency pause to prevent its users from initiating new trades and deposits in case of emergencies. In the event of an emergency, users can continue to withdraw their assets via multi-asset withdrawals.

Our team investigated the implementation of the emergency pause mechanism, which gives the admin control over security-critical functionalities. We examined the arithmetic operations performed by the smart contracts as well as the handling of high precision tokens. We checked the functionality for

removing liquidity from the pool. We also checked the appropriate use of reentrancy safeguards and the mechanisms implemented to mitigate front running and did not identify any vulnerabilities.

Code Quality

The Metapool smart contracts, and the Saddle Finance smart contracts in general, are well organized and modular and generally adhere to best practices. We found that the `MetaSwap`, `MetaSwapDeposit` and `MetaSwapUtils` smart contracts use an outdated Solidity compiler. We recommend that a more recent version of the compiler be used (Suggestion 1).

Tests

Our team found that the Metapool smart contracts have sufficient test coverage that includes success, failure, and some edge case scenarios. We suggest an improvement to make using the tests more intuitive (Suggestion 2).

Documentation

The Saddle Finance user documentation is comprehensive and provides a helpful and accurate description of the entire system. This documentation also transparently describes potential risks to the system and its users, in addition to providing non-technical examples for use cases as well as screenshots of user interfaces. However, we found that the documentation specific to the Metapool smart contracts should be improved. In order to get a better understanding, our team had to rely on the documentation of the base pool smart contracts and cross-reference it against components that were out of scope. We recommend that the documentation of the Metapool smart contracts be improved (Suggestion 3).

Code Comments

The codebase of the Metapool smart contracts implements sufficient code comments, which describe the intended behavior of critical functions in the smart contracts, and the code comments generally adhere to the Solidity NatSpec guidelines.

Scope

The scope of this review was limited to a single component within a larger system of smart contracts. The scope was sufficient to check for implementation errors, security vulnerabilities, and adherence to best practice within the smart contracts reviewed. However, our team recommends the continued research of the security of the system at large.

Specific Suggestions

We list the suggestions found during the review, in the order we reported them. In most cases, remediation of an issue is preferable, but mitigation is suggested as another option for cases where a trade-off could be required.

SUGGESTION	STATUS
Suggestion 1: Upgrade Solidity Version	Resolved
Suggestion 2: Move Plain Values to Variables in Tests (Out of Scope)	Resolved
Suggestion 3: Improve Documentation of Metapool Smart Contracts	Resolved

Suggestions

Suggestion 1: Upgrade Solidity Version

Location

MetaSwap.sol#L3

MetaSwapDeposit.sol#L3

MetaSwapUtils.sol#L3

Synopsis

The pragma on the contracts is version 0.6.12, while the latest is 0.8.15 (excluding nightly builds). The external libraries offer support for 0.8.x or are no longer needed, so there is no compatibility issue. Version 0.6.x allows the use of unsafe features that were removed in newer versions. This allows for a broader attack surface.

Mitigation

We recommend upgrading to the newest compiler version, as it may include features and bug fixes that were present in previous versions.

Status

The Saddle Finance team has created new contracts with the new version of Solidity (version 0.8.17). .

Verification

Resolved.

Suggestion 2: Move Plain Values to Variables in Tests (Out of Scope)

Location

Examples (non-exhaustive):

test/metaSwap.ts

test/metaSwapDecimals.ts

test/metaSwapDeposit.ts

Synopsis

Some number values are used in more than one place in the tests for assertions. They are used as plain values. However, it is best practice to create constants with intuitive names, which can be used more consistently in those tests to increase readability and make the tests less prone to errors.

Mitigation

We recommend using constants for values in the tests for assertions.

Status

The Saddle Finance team has declared constants with intuitive names as suggested.

Verification

Resolved.

Suggestion 3: Improve Documentation of Metapool Smart Contracts

Location

https://docs.saddle.finance/solidity-docs/swap

Synopsis

Saddle Finance has sufficient user documentation describing the general architecture, interaction, and potential security risks. In addition, some of the smart contracts are well documented. However, the Metapool smart contracts do not have the same degree of documentation as the base pool smart contracts. This required our team to cross-reference the Metapool smart contracts against the base pool smart contracts and documentation. This can be confusing and time-consuming to users and security researchers.

Mitigation

We recommend that the documentation of the Metapool smart contracts be improved to the level of the documentation present in the base pool smart contracts.

Status

The Saddle Finance team has added additional documentation for the `MetaSwap.sol`, `MetaSwapDeposit.sol`, and `MetaSwapUtils.sol` contracts.

Verification

Resolved.

About Least Authority

We believe that people have a fundamental right to privacy and that the use of secure solutions enables people to more freely use the Internet and other connected technologies. We provide security consulting services to help others make their solutions more resistant to unauthorized access to data and unintended manipulation of the system. We support teams from the design phase through the production launch and after.

The Least Authority team has skills for reviewing code in multiple Languages, such as C, C++, Python, Haskell, Rust, Node.js, Solidity, Go, JavaScript, ZoKrates, and circom, for common security vulnerabilities and specific attack vectors. The team has reviewed implementations of cryptographic protocols and distributed system architecture in cryptocurrency, blockchains, payments, smart contracts, and zero-knowledge protocols. Additionally, the team can utilize various tools to scan code and networks and build custom tools as necessary.

Least Authority was formed in 2011 to create and further empower freedom-compatible technologies. We moved the company to Berlin in 2016 and continue to expand our efforts. We are an international team that believes we can have a significant impact on the world by being transparent and open about the work we do.

For more information about our security consulting, please visit https://leastauthority.com/security-consulting/.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis

Our audit techniques include manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's website to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. As we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation. We hypothesize what vulnerabilities may be present and possibly resulting in Issue entries, then for each, we follow the following Issue Investigation and Remediation process.

Documenting Results

We follow a conservative and transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even before having verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this, we analyze the feasibility of an attack in a live system.

Suggested Solutions

We search for immediate and comprehensive mitigations that live deployments can take, and finally, we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our Initial Audit Report, and before we perform a verification review.

Before our report, including any details about our findings and the solutions are shared, we like to work with your team to find reasonable outcomes that can be addressed as soon as possible without an overly negative impact on pre-existing plans. Although the handling of issues must be done on a case-by-case basis, we always like to agree on a timeline for a resolution that balances the impact on the users and the needs of your project team.

Resolutions & Publishing

Once the findings are comprehensively addressed, we complete a verification review to assess that the issues and suggestions are sufficiently addressed. When this analysis is completed, we update the report and provide a Final Audit Report that can be published in whole. If there are critical unaddressed issues, we suggest the report not be published and the users and other stakeholders be alerted of the impact. We encourage that all findings be dealt with and the Final Audit Report be shared publicly for the transparency of efforts and the advancement of security learnings within the industry.