DSF security review

2023-09-03

DSF security review

Executive summary

This document represents an audit for DSF, which is a DeFi platform operating on the Ethereum blockchain.

The document contains the following chapters:

1. *Introduction*: a high-level overview of the project and the methodology of the audit
2. *Technical overview*: in-depth overview of the project from a security perspective
3. *Risk assessment*: an overview of the threats and risks that considers the underlying protocols
4. *Security code review*: security analysis of the DSF source code
5. *Suggestions*: recommendations and suggestions on how the security of the project can be improved

1 Introduction

DSF, which stands for Defining Successful Future, is a DeFi (Decentralized Finance) platform operating on the Ethereum blockchain. The platform's primary value proposition is simplifying access to DeFi financial products with a user-friendly interface.

The current implementation includes the platform for investments: Users can invest in Stablecoins, Ethereum (ETH), or Wrapped Bitcoin (WBTC) through DSF. The platform emphasizes the use of reliable tools, including decentralized exchanges and yield optimizers.

The project is written in Solidity. The source code is stored in the following repo: https://github.com/Dsf-Finance/Investments. The Mainnet address of the contract is 0x22586ea4fdaa9ef012581109b336f0124530ae69.

The audit was performed by Georgiy Komarov at September 2023. The audited version has commit number 28a70094314065c3ff991421be43b880ceb9ed77.

The audit methodology comprises the following steps:

1. Collecting and analyzing information about the project
2. Security source code review, which includes the following phases:
   - Understanding the project's business logic
   - Running static analyzers
   - Conducting a manual security code review
3. Analyzing threats and project-specific risks, which involves:
   - analyzing the security aspects of the projects that DSF relies on
   - listing and assessing potential threats to the entire system
4. Publishing the audit findings

The following tools were used when auditing the project:

- quint to model and understand some parts of the source code
- a custom tool to inspect the structure of the project and visualize it
- slither which is a static analyzer for Solidity
- semgrep patterns for common Solidity problems

Scope of the audit

DSF is implemented as a composable DeFi application. Therefore, any security issues in one of the underlying protocols can affect DSF as well, so this must be carefully considered by potential users. However, given the complexity of reviewing all underlying details in a single document, we provide only a high-level overview of these projects and consider the possible risks for DSF users.

Taking into account all the previous details, the scope of the audit is defined as follows:

- the source code of the DSF contracts uploaded to Mainnet is thoroughly reviewed
- all the risks related to the underlying protocols are included in the Risk Assessment section

The following components of the system are outside the scope:

- the infrastructure used to run the project
- the front-end part of the application and its communication with contracts

2 Technical overview

Overview

DSF Invest is a DeFi that provides access to LP pools. Currently, it is implemented on top of Abracadabra Money, which uses Convex LP pools, but its architecture allows to extend the list of the supported LP pool providers. In smart-contracts development this approach is called composable DeFi, which means, that a contract on Ethereum might reuse the existing contracts to create something new.

The main idea of DSF Invest is the following: it accumulates user's tokens and sends them to LP pools. The value of the contract for users is the following:

- It provides a simple UI to LP pools
- Using it allows the user to minimize funds used as gas to interaction with LP pools

The following diagram shows the interaction of DSF with the underlying DeFi projects:

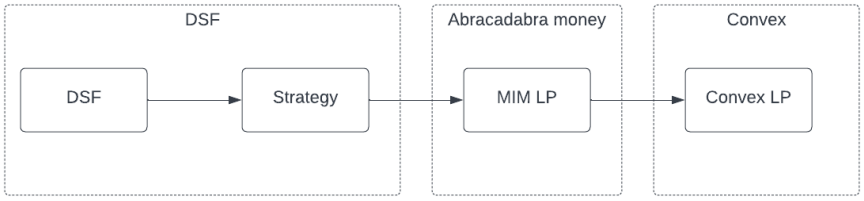

Figure 1: diagram

DSF is the main contract. It provides the functions used by the user to communicate with different investment strategies, and functionality to administer the contract. *Strategy* in terms of DSF means an interface to communicate with LP pools. Currently it contains a few interfaces and the single implementation to work with SPELL LP pools: the *ConvexStratedgy_MIM_SPELL* contract.

3 Security code review

The following section contains the description of potential problems in the source code of the DSF contracts and mitigation suggestions.

(HIGH) Curve read-only reentrancy vulnerability in `CurveConvexStrat`

There are a few so called read-only reentrancy vulnerabilities that previously affected a few Curve pools. These are related to using of the `get_virtual_price` function, potentially allowing malicious manipulation of LP token prices in various pools. The affected projects included MakerDAO, Enzyme, Abracadabra, TribeDAO, and Opyn.

Currently, DSF implements some logic to call this function, but it doesn't use any Curve pools. Therefore there is no exploitation scenario. Despite that, the severity of this issue should be considered as High, since it may lead in direct losing of user funds in some scenarios, depending on how these pools will be integrated and used in the future.

These possible security implications should be kept in mind in the development,

and it makes sense to apply the suggested mitigation with function locks described in the report by ChainSecurity.

(HIGH) No slippage check for Uniswap' `swapExactTokensForTokens`

Slippage is a common problem in decentralized exchanges. The point is that the price which the user sees when initiating the transaction may be changed during the transaction processing.

There are a few uses of `swapExactTokensForTokens` in the strategy interfaces. The both uses in `sellRewards` should be considered as safe, since this is an internal function called only by operator of the contract.

But the slippage problem may appear in the `sellExtraToken` function, since it is a public function. The suggested mitigation is to set the `amountOutMin` parameter of `swapExactTokensForTokens` or change the visibility of `sellExtraToken`.

(MEDIUM) Improper input validation in public functions

There are a few cases in the source code where the contract doesn't check properly incorrect inputs. These cases should not be considered as vulnerabilities as is, but they may lead to losing user funds indirectly, if the bugs in the frontend are present. Therefore, input arguments of all the public functions need to be extended with the sanity checks, making sure that the user is not trying to send their funds to zero address or to its own account.

(LOW) Use the latest `solc` version without floating pragma

Avoid using floating pragma: `pragma solidity ^0.8.0;`. This is considered as a vulnerability (SWC103), since contracts should be deployed with the version they were tested.

(INFO) Remove magic numbers and document the source more clearly

Having more documentation comments for strategies and DSF function will be helpful in the long-term maintaining and developing of the project.

(INFO) Size of the main contract should be reduced

According to EIP-170, there is a limitation of the contract size that could be uploaded to Mainnet. DSF almost reaches that limitation, and that might be a problem in the future.

Therefore, it is suggested to change the architecture of the contract, making it more modular and reducing its size.

(INFO) Automate deploying contract to Mainnet

The process of deployment the contract should be automated, using Truffle or Hardhat. This is important, since the initialization process of DSF includes configuration of different addresses, which makes this process potentially erroneous.

4 Risk assessment

The Composable DeFi architecture has pros and cons from a security perspective. In general, these applications tend to have a higher level of security because they leverage existing contracts rather than developing code from scratch, so they are less likely will introduce their own vulnerabilities. Despite that, it is important to understand what the application is using under the hood, and what is the security properties of these contracts, because any vulnerability in the underlying contracts will affect the user contract as well.

Another common threat for DSF is typical for DApps and corresponds to keys management. If an attacker get an access to private keys used to administer or operate the contract, they will be able to make a critical damage to the project, including stealing user funds.

Taking the above into account, we define possible threats to the DSF project that, when implemented, may result in a single risk: the direct losing of user funds. The following table summarizes these risks:

Threat	Description
1 Implementation errors	Implementation errors in the contract may lead to losing funds that are currently stored in the DSF contract. These funds are regularly deposited to the configured LP pools.
2 Infrastructure misconfiguration	Misconfiguration of the underlying infrastructure used to operate the system can lead to losing control to the contract on Operator's role.
3 Improper keys management	Losing keys to the contract administrator or operator will lead to losing all the funds currently stored in the contract, which is possible if the attacker upgrade logic of some DSF contracts.
4 Hack or Exit Scam of Abracadabra Money (MIM)	Consider the following facts about MIM:

Threat	Description
	* the team is anonymous * currently no bug-bounty program is present * no audits expect an "anonymous" one * previously have depeg history their users Considering all the above, depending on that project makes a security risk to DSF.

Summarizing the described risks above, potential users should understand that the safety of their tokens depends on the underlying protocols, which currently include Abracadabra Money and Convex. The risks of using DSF should be regarded in the same as the risks associated with any other liquidity pool.

To mitigate these risks, users should conduct thorough research, stay informed about protocol updates, and consider risk management strategies. Remember that using this protocol offers the potential for higher income compared to more conservative DeFi options like lending protocols. However, it also introduces additional risks inherent to systems of this nature.

5 Suggestions

This section contains some high-level suggestions related to the implementation and architecture of DSF.

5.1 Apply the suggested fixes from Section 3

The "Security code review" section contains some suggestions regarding the source code of the contracts.

5.2 Understand and diversify the underlying protocols

The `DSF` contract uses generic interfaces for `Strategy`, therefore it is possible to use any DeFi that supports Curve or Convex interfaces. Following this approach allows to diversify the underlying logic and reduce risks corresponding to hacks of the specific protocol.

Therefore, it is important to choose a few suitable DeFi systems, and thoroughly research their security properties, including the previous history, audits, the presence of bug-bounty program. Then a few projects should be chosen instead Abracadabra Money to improve the security properties of the system.

5.3 Apply security best practices for infrastructure and keys management

When using centralized infrastructure to perform operations on DSF, it is important to ensure the security of these processes. These topics are out of the scope of this review, and they should be considered independently.

5.4 Introduce the bug-bounty program

A well-executed bug bounty program can enhance the reputation and trustworthiness of a project. Demonstrating a commitment to security and transparency can attract more users and investors who have confidence in the platform's integrity.

5.5 Share information about the team publicly

When users and investors can easily verify the credentials and backgrounds of team members, it adds credibility to the project. It reassures them that the project is led by competent and experienced individuals who are capable of delivering on their promises. Therefore, it is a good idea to add an information about the team to the website, and to use real accounts in the development on Github.

5.6 Apply best practices of development

As a safety-critical application, smart contracts contracts should undergo thorough reviews before being uploaded to the mainnet to the greatest extent possible.

It is suggested to:

- Include unit tests for critical parts of the source code
- Setup the CI (Continuous Integration) system that calls different SAST and DAST tools for Solidity
- Conduct security code review of new major releases before uploading it to Mainnet