DeFi Plaza
Security Analysis

by Pessimistic

This report is public.

Published: September 24, 2021

Abstract

In this report, we consider the security of smart contracts of DeFi Plaza project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, security audit is not an investment advice.

Summary

In this report, we considered the security of DeFi Plaza smart contracts. We performed our audit according to the procedure described below.

The initial audit showed several issues of medium severity, including Bug, ERC20 standard violation, and Overpowered owner, and several low-severity issues.

The project has insufficient documentation. Some functionality was not covered with tests.

After the initial audit, the code base was updated. In this update, most of the issues were fixed. Also, new tests were added to the project.

General recommendations

We recommend adding detailed documentation.

Project overview

Project description

For the audit, we were provided with DeFi Plaza project on a private GitHub repository, commit 2644b7106e36b646194c0b5b27cbd61e749bc764.

The scope of the audit included only files:

- **DeFiPlaza.sol**
- **DFPgovernance.sol**

The project has no documentation. However, the code itself is thoroughly documented.

The project compiles without any issues. All 91 tests pass, the coverage is 96%. However, some functionality is not covered.

The total LOC of audited sources is 638.

Code base update

After the initial audit, the code base was updated. For the recheck, we were provided with commit de0c91f09f92a3743df8f7a8cd3605523ad6c4f6.

In this update, most of the issues were fixed. Also, new tests were added, the overall coverage reached 100%.

Procedure

In our audit, we consider the following crucial features of the code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Automated analysis
  - We scan project's code base with automated tools: Slither and SmartCheck.
  - We manually verify (reject or confirm) all the issues found by tools.
- Manual audit
  - We manually analyze code base for security vulnerabilities.
  - We assess overall project structure and quality.
- Report
  - We reflect all the gathered information in the report.

Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger smart contracts security. We highly recommend fixing them.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence project operation in current implementation. We highly recommend addressing them.

Bug (fixed)

In **DFPgov** contract, `stopProgram()` function is supposed to set `state.startTime` to its maximum value. However, at line 274 it is assigned `0xffff` instead of `0xffffffff`. As a result, all `block.timestamp >= state.startTime` checks pass and reward calculation is affected.

Consider using `type(uint32).max` syntax to improve code readability and avoid confusion.

The issue has been fixed and is not present in the latest version of the code.

ERC20 standard violation (fixed)

EIP-20 states:

```
Callers MUST handle false from returns (bool success). Callers MUST NOT
assume that false is never returned!
```

However, the returned value of `transfer()` call is not checked in

- **DeFiPlaza** contract at line 465.
- **DFPgov** contract at line 145.

The issues have been fixed and are not present in the latest version of the code.

Insufficient documentation

The project has insufficient documentation. The code base has detailed comments. However, they do not explain the whole structure of the project and the intention of certain functions. Thus, it is sometimes unclear whether the behavior of the code is correct.

The documentation is critically important not only for the audit but also for development process. It should explicitly explain the purpose and behavior of the contracts, their interactions, and main design choices.

Low severity issues

Low severity issues can influence project operation in future versions of code. We recommend taking them into account.

Code quality

- Consider using time units for time interval constants to improve code readability, e.g. `365 days` instead of `31536000` in **DFPgov** contract.

  The issue has been fixed and is not present in the latest version of the code.

- In `addMultiple()` function of **DeFiPlaza** contract, variable `previous` is assigned a new value at line 263, but this value is not used.

  The issue has been fixed and is not present in the latest version of the code.

- Consider adding detailed comments in `addLiquidity()` function of **DeFiPlaza** contract for numerical calculation of $(1 + x)^{1/16}$ with binomial series at lines 190–200.

  The explaining comments have been added to the code.

- `bootstrapNewToken()` function of **DeFiPlaza** contract allows an underflow for no reason. Consider checking the value or documenting this behavior explicitly.

  The explaining comment has been added to the code.

- In **DeFiPlaza** contract, functions `setDeListingBonus()`, `setTradingFee()`, and `setAdmin()` modify configuration parameters of the contract and therefore should emit events.

  > *Comment from developers: Would be cleaner but adds costs for deployment and execution. Decided to save that.*

- In **DFPgov** contract, the code for staking state update is repeated several times. Consider moving it to internal function.

- Consider using proper NatSpec format in the code.

Notes

Overpowered owner

The owner of **DeFiPlaza** contract can change listing with any tokens and then swap them with other tokens within one transaction so that other users will be unable to react. The owner can also change trading fee, set delisting bonus, etc.

In the current implementation, the system depends heavily on the owner of the contract. There are scenarios that may result in undesirable consequences for the project and its users, e.g., if the owner's private keys become compromised.

We recommend designing contracts in a trustless manner or using proper key management system, e.g., multisig.

> *Comment from developers: This is by design, the owner will be set to the `timeLock` contract, for which the only proposer will be a multisig wallet.*

Slippage protection

Note that in `addLiquidity()` function of **DeFiPlaza** contract, for excessive liquidity, the growth rate of slippage increases significantly. Therefore, slippage protection is highly recommended. For larger amounts, `addMultiple()` is preferable.

Additional comments with slippage issue description have been added to the function.

This analysis was performed by Pessimistic:

Vladimir Tarasov, Security Engineer
Evgeny Marchenko, Senior Security Engineer
Boris Nikashin, Analyst
Irina Vikhareva, Project Manager

September 24, 2021