SMART CONTRACT AUDIT REPORT

# SparkDEX

Version: 1.1

Protofire
September, 2024

# Table Of Contents

# The SparkDEX audit report

The audit report was prepared for the **SparkDEX team** and includes following Github repository:

https://github.com/SparkDEX/v3-core/ audited commits:

1. 3fe076ede3a9a3e75839a842fadc5892fe53dce5
2. 693a06cc3b23c16854bfdeb16543e338ab463efc
3. c43de28bc618688badf2b5653569d3b52d3b05b0
4. fe10122877ee4ae6cae882ef359dcbd8095fa3d3

The code repository does not contain any tests.

## Procedure

The audit includes the following procedures:

- code analysis by the Slither static analyzer, followed by manual analysis and selection of problems
- manual code analysis, including logic check
- checking the business logic for compliance with the documentation (or white paper)

In addition, during the audit, we also provide recommendations on optimizing smart contracts and using best practices.

## Summary of audit findings

| Severity | Count |
|---|---|
| HIGH | 0 |
| MEDIUM | 1 |
| LOW | 0 |
| INFORMATIONAL | 1 |
| **TOTAL** | **2** |

## Severity classification

**High severity issues**

High severity issues can lead to a direct or indirect loss of funds by both users and owners, a serious violation of the logic of the contract, including through the intervention of third parties. Such issues require immediate correction.

**Medium severity issues**

Medium severity issues may cause contract functionality to fail or behave incorrectly. Also, medium severity issues can cause financial damage. Such issues require increased attention.

**Low severity issues**

Low severity issues do not have a major security impact. It is recommended that such issues be taken into account.

**Informational severity issues**

These issues provide general guidelines for writing code as well as best practices.

## Smart contract UniswapV3Pool [SDX-01]

A Uniswap pool facilitates swapping and automated market making between any two assets that strictly conform to the ERC20 specification.

The original contract code was supplemented with the function functionCall() to enable virtual delegation (voting, claiming rewards, etc.) in other protocols using the pool's token balance. The development team implemented restrictions on the use of pool tokens (`require(target != token0 && target != token1)`) to prevent them from being withdrawn or approved for subsequent withdrawal.

Due to the contract size limitation, the development team had to remove some rarely used events (`SetFeeProtocol`, `CollectProtocol`). In our view, this does not significantly degrade the contract code and does not affect its security.

| ID: **SDX01-01** | Severity: **Medium** | Status: **Fixed** |
|---|---|---|

### Balance check

Despite the presence of the check:
``require(target != token0 && target != token1)``
the audit team strongly recommends adding balance validation after executing the ``functionCall()`` to minimize risks for liquidity providers.

It can be achieved by adding a require statement without any message errors, and using an optimization config with values < 56.

|  |  |  |
|---|---|---|
| ID: **SDX01-02** | Severity: **Info** | Status: **Acknowledged** |

## Call to an EOA account

In the function `functionCall()` there is a call to the address `target`.

If this address is an EOA account, this call won’t be reverted, this may introduce some delusions about this call.

## Smart contract UniswapV3Factory [SDX-02]

The UniswapV3Factory contract deploys Uniswap V3 pools and manages ownership and control over pool protocol fees.

The original contract code was slightly modified. Due to the contract size limitation, the development team had to remove some events (`FeeAmountEnabled`, `OwnerChanged`). Since these events were removed only for well-known fee values, we believe this does not significantly degrade the contract code and does not affect its security.

**No issues have been identified.**

## Conclusion

The SparkDEX project utilizes a well-known fork of Uniswap-v3-core. The development team has added an interesting feature that significantly enhances the pool's potential for interacting with external projects. This feature can be used for voting, claiming rewards and etc.

One medium severity issue were found during the audit. We highly recommend fixing it.

Please note that auditing is not a complete replacement for unit tests and integration tests.

**Conclusion Update.** The issue has been resolved by the dev team in the code update.

## Disclaimer

Note that smart contract audit provided by Protofire is not designed to replace functional tests required before any software release, and does not give any warranties on finding all possible security issues of the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues. As one audit-based assessment cannot be considered comprehensive, Protofire recommends proceeding with several independent audits and a public bug bounty program to ensure the security of smart contract(s). Smart contract audit provided by Protofire shall not be used as investment or financial advice.