VIDMA

XEND Finance

SMART CONTRACT AUDIT

Project: Xend Finance
Date: May 7th, 2022

TABLE OF CONTENTS

SUMMARY

Vidma is pleased to present this audit report outlining our assessment of code, smart contracts, and other important audit insights and suggestions for management, developers, and users.

Over the course of the audit, we identified certain misbehaviors that can manifest as well best practices that can be applied to the codebase to further enhance the security level it attains.

Additionally, The contract is highly dependent on the returned data from the BE and the work of third-party aggregators which was not in the scope of the audit. A thorough line-by-line review was conducted on the codebase to identify potential malfunctions and vulnerabilities. Intricate care was put into ensuring that the flow of funds within the system conforms to the specifications and restrictions laid forth within the protocol's specification.

All of the found issues were successfully resolved by the Xend Finance team.

During the audit process, the Vidma team has found several issues with different severity, but no found issues with critical severity. A detailed summary and the current state are displayed in the table below.

Severity of the issue	Total found	Resolved	Unresolved
Critical	0 issues	0 issues	0 issues
High	1 issue	1 issue	0 issues
Medium	2 issues	2 issues	0 issues
Low	3 issues	3 issues	0 issues
Informational	2 issues	2 issues	0 issues
Total	**8 issues**	**8 issues**	**0 issues**

After evaluating the findings in this report and the final state after fixes, the Vidma auditors can state that the contract is fully operational and secure. Under the given circumstances, we set the following risk level:

Our auditors are evaluating the initial commit given for the scope of the audit and the last commit with the fixes. This approach helps us adequately and sequentially evaluate the quality of the code. Code style, optimization of the contracts, the number of issues, and risk level of the issues are all taken into consideration. The Vidma team has developed a transparent scoring system presented below.

Severity of the issue	Resolved	Unresolved
Critical	1	10
High	0.8	7
Medium	0.5	5
Low	0.2	0.5
Informational	0	0.1

Please note that the points are deducted out of 100 for each and every issue on the list of findings (according to the current status of the issue). Issues marked as "not valid" are not subject to point deduction.

Based on the **overall result of the audit**, the Vidma audit team grants the following score:

In addition to manual check and static analysis, the auditing team has conducted a number of integrated autotests to ensure the given codebase has an adequate performance and security level.

The test results and the coverage can be found in the accompanying section of this audit report.

Please be aware that this audit does not certify the definitive reliability and security level of the contract. This document describes all vulnerabilities, typos, performance issues, and security issues found by the Vidma audit team. If the code is still under development, we highly recommend running one more audit once the code is finalized.

SCOPE OF WORK

XEND Finance

Credit Unions, Cooperatives, and Individuals anywhere in the world can now earn higher interests in stable currencies on their savings.

Within the scope of this audit, two independent auditors thoroughly investigated the given codebase and analyzed the overall security and performance of the smart contracts.

The audit was conducted from April 1st, 2022 to May 7th, 2022. The outcome is disclosed in this document.

The scope of work for the given audit consists of the following contracts:

- SwapRouter.

The source code was taken from the following **source**:
https://github.com/xendfinance/MadWalletDEXAggregator

Initial commit submitted for the audit:
blob/main/contracts/SwapRouter.sol

Last commit reviewed by the auditing team:
894f0e0679afec453a477a223a0e8e3f776d482d

As a reference to the contracts logic, business concept, and the expected behavior of the codebase, the Xend Finance team has provided the following documentation:
https://github.com/xendfinance/MadWalletDEXAggregator/tree/swap/router#readme
https://stake.xend.tools/networks/56/trades?destinationToken=&sourceToken=&sourceAmount=&slippage=&timeout=&walletAddress=

WORKFLOW OF THE AUDITING PROCESS

Vidma audit team uses the most sophisticated and contemporary methods and well-developed techniques to ensure contract is free of vulnerabilities and security risks. The overall workflow consists of the following phases:

Phase 1: The research phase

Research

After the Audit kick-off, our security team conducts research on the contract's logic and expected behavior of the audited contract.

Documentation reading

Vidma auditors do a deep dive into your tech documentation with the aim of discovering all the behavior patterns of your codebase and analyzing the potential audit and testing scenarios.

The outcome

At this point, the Vidma auditors are ready to kick off the process. We set the auditing strategies and methods and are prepared to conduct the first audit part.

Phase 2: Manual part of the audit

Manual check

During the manual phase of the audit, the Vidma team manually looks through the code in order to find any security issues, typos, or discrepancies with the logic of the contract. The initial commit as stated in the agreement is taken into consideration.

Static analysis check

Static analysis tools are used to find any other vulnerabilities in smart contracts that were missed after a manual check.

The outcome

An interim report with the list of issues.

Phase 3: Testing part of the audit

Integration tests

Within the testing part, Vidma auditors run integration tests using the Truffle or Hardhat testing framework. The test coverage and the test results are inserted in the accompanying section of this audit report.

The outcome

Second interim report with the list of new issues found during the testing part of the audit process.

STRUCTURE AND ORGANIZATION OF THE FINDINGS

For simplicity in reviewing the findings in this report, Vidma auditors classify the findings in accordance with the severity level of the issues. (from most critical to least critical).

All issues are marked as “Resolved” or “Unresolved”, depending on if they have been fixed by Xend Finance or not. The issues with “Not Valid” status are left on the list of findings but are not eligible for the score points deduction.

The latest commit with the fixes reviewed by the auditors is indicated in the “Scope of Work” section of the report.

The Vidma team always provides a detailed description of the issues and recommendations on how to fix them.

Classification of found issues is graded according to 6 levels of severity described below:

Critical

The issue affects the contract in such a way that funds may be lost or allocated incorrectly, or the issue could result in a significant loss.

Example: Underflow/overflow, precisions, locked funds.

High

The issue significantly affects the ability of the contract to compile or operate. These are potential security or operational issues.

Example: Compilation errors, pausing/unpausing of some functionality, a random value, recursion, the logic that can use all gas from block (too many iterations in the loop), no limitations for locking period, cooldown, arithmetic errors which can cause underflow, etc.

Medium

The issue slightly impacts the contract's ability to operate by slightly hindering its intended behavior.

Example: Absence of emergency withdrawal of funds, using assert for parameter sanitization.

Low

The issue doesn't contain operational or security risks, but are more related to optimization of the codebase.

Example: Unused variables, inappropriate function visibility (public instead of external), useless importing of SCs, misuse or disuse of constant and immutable, absent indexing of parameters in events, absent events to track important state changes, absence of getters for important variables, usage of string as a key instead of a hash, etc.

Informational

Are classified as every point that increases onboarding time and code reading, as well as the issues which have no impact on the contract's ability to operate.

Example: Code style, NatSpec, typos, license, refactoring, naming convention (or unclear naming), layout order, functions order, lack of any type of documentation.

MANUAL REPORT

Possible broke of contract by upgradable dependency

High | Resolved

Contract SwapRouter dependence on pre-deployed proxy contracts paraswapRouter.

0x: Exchange Proxy. It means that this proxy contract can be updated and the logic of swap can be broken. In that case SwapRouter won't be able to swap with use of those contracts.

Recommendation:

Implement proxy pattern for SwapRouter to be able to fix if any changes will be in external proxy contracts.

Unchecked tokens transfer

Medium | Resolved

Function *swap()* ignores return value by token transfer at lines 161, 301, 311.

Recommendation:

Use SafeERC20, or ensure that the *transfer/transferFrom* return value is checked.

Unchecked BNB transfer

Medium | Resolved

Function *swap()* ignores return value by BNB transfer at lines 304, 317.

Recommendation:

Add check for return value.

Floating pragma

Low | Resolved

The current version of solc in contracts SwapRouter is ^0.8.0 and it is better to lock the pragma to a specific version.

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommendation:

Lock pragma to a specific version (0.8.9).

Missing Zero Address Validation

Low | Resolved

In function *updateFeeAddress()* no zero address validation.

Recommendation:

Check that the address is not zero.

Inappropriate function visibility

Low | Resolved

Functions *updateFeeAddress()* and *swap()* are declared as public but they are never used inside of the contract.

Recommendation:

Change function visibility to external.

Lack of NatSpec annotations

Informational | Resolved

Smart contracts SwapRouter is not covered by NatSpec annotations.

Recommendation:

Consider covering by NatSpec all contract methods.

Unused commented functionality

Informational | Resolved

In function *getParaswapData_2()* at line 410, in function *getPmmOrderInfo_1()* at line 469, in function *getOneInchDescData_1()* at line 545.

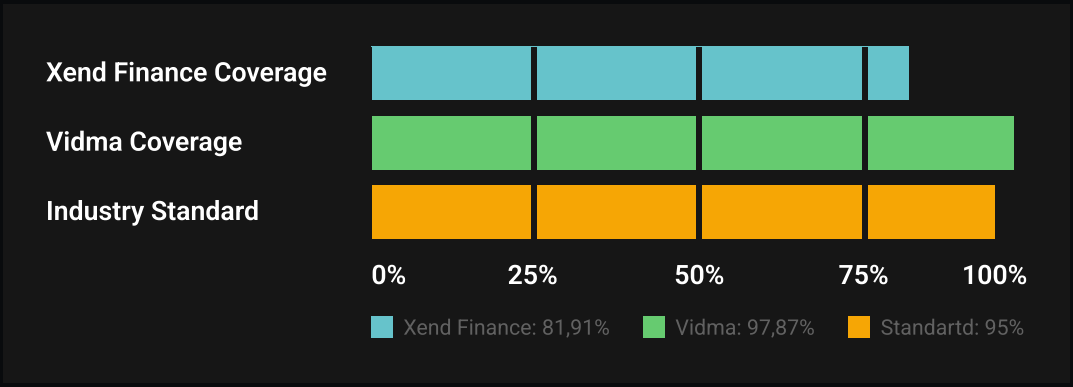
Recommendation:

Remove commented functionality.

TEST RESULTS

To verify the security of the contract and the performance, a number of integration tests were carried out using the Truffle testing framework.

In this section, we provide both tests written by Xend Finance and tests written by Vidma auditors.



It is important to note that Vidma auditors do not modify, edit or add tests to the existing tests provided in the Xend Finance repository. We write totally separate tests with code coverage of a minimum of 95% to meet the industry standards.

Tests written by Xend Finance

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	81.91	50	90.91	82.14
swapRouter.sol	81.91	50	90.91	82.14
All Files	**81.91**	**50**	**90.91**	**82.14**

Test Results

```
Contract: test Test
  balance: 1868956732697112000000
  balance: 0
  balance: 47896547748258
  balance: 94973429280000000000
  ✓ test (11148ms)

Contract: test Test
  balance: 21321543327
  balance: 0
  balance: 175000000004250
  balance: 99969535159999500000
  ✓ test (4851ms)

Contract: test Test
  balance: 170721419513686443107
  balance: 0
  balance: 554547097092072
  balance: 99972332036000000000
  ✓ test (16287ms)

Contract: test Test
  balance: 14131594759140173090
```

```
    balance: 262500000000000
    balance: 99961810540000000000
    ✓ test (5552ms)

  4 passing (41s)
```

Tests written by Vidma

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	97.87	79.63	100.00	96.33
SwapRouter.sol	97.87	79.63	100.00	96.33
All Files	**97.87**	**79.63**	**100.00**	**96.33**

Test Results

```
Contract: SwapRouter
  initialization
    ✓ should initialize aggregators addresses correct (465ms)
    ✓ should initialize fee receiver address correct (120ms)
  updateFeeAddress
    ✓ should update fee address correct (572ms)
    ✓ shouldn't set fee address as zero address (722ms)
  paraswapRouter
    should swap on paraswap router correct
      ✓ swap ETH for ERC20 (6244ms)
      ✓ swap ERC20 for ETH (5918ms)
      ✓ swap ERC20 for ERC20 (6059ms)
  airswapLight
    should swap on airswapLight router correct
      ✓ swap ERC20 for ERC20 (6731ms)
  zeroExRouter
    should swap on zeroEx router correct
      ✓ swap ETH for ERC20 (9533ms)
      ✓ swap ERC20 for ETH (7532ms)
      ✓ swap ERC20 for ERC20 (4546ms)
  oneInchRouter
    should swap on oneInch router correct
```

```
    ✓ swap ETH for ERC20 (3664ms)
    ✓ swap ERC20 for ETH (5583ms)
    ✓ swap ERC20 for ERC20 (5527ms)

  14 passing (1m)
```

We are delighted to have a chance to work with the Xend Finance team and contribute to your company's success by reviewing and certifying the security of your smart contracts

The statements made in this document should be interpreted neither as investment or legal advice, nor should its authors be held accountable for decisions made based on this document.

Vidma is a security audit company helping crypto companies ensure their code and products operate safely and as intended, enabling founders to sleep soundly at night. We specialize in auditing DeFi protocols, layer one protocols, and marketplace solutions. Our team consists of experienced and internationally trained specialists. Our company is based in Ukraine, known for its strong engineering, cryptography, and cybersecurity culture.

Website: **vidma.io**
Email: **security@vidma.io**