VIDMA

XEND Finance

SMART CONTRACT AUDIT

Project: Xend Finance
Date: April 11th, 2022

TABLE OF CONTENTS

SUMMARY

Vidma team has conducted a security audit of the smart contracts for the given codebase. After a thorough check and analysis, we can state that the contacts are in good condition. All the issues found during the manual and testing parts of the audits are fixed (1 issue is marked as “Invalid”). Based on the result of the fixes, Vidma auditing team can state that the contacts are safe to use and are ready for production.

A detailed summary of the issues and their current state is displayed in the table below.

Severity of the issue	Total found	Resolved	Invalid	Unresolved
Critical	0 issues	0 issues	0 issues	0 issues
High	4 issues	3 issues	1 issues	0 issues
Medium	1 issues	1 issues	0 issues	0 issues
Low	6 issues	6 issues	0 issues	0 issues
Informational	2 issues	2 issues	0 issues	0 issues
Total	**13 issues**	**12 issues**	**1 issues**	**0 issues**

Under the given circumstances we can set the following risk level:

Vidma auditors are evaluating the initial commit given for the scope of the audit and the last commit with the fixes. Hence, it helps to adequately evaluate the development quality. Code style, optimization of the contracts, amount and risk level of the issues are taken into consideration. The Vidma team has developed the 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

Based on the given findings, risk level, performance, and code style, Vidma team can grant the following overall score:

Vidma auditing team has conducted a bunch of integrated autotests to ensure that the given codebase has decent performance and security levels.
The test results and the coverage can be found in the accompanying section of this audit report.

Please mind that this audit does not certify the definite reliability and security level of the contract. This document describes all vulnerabilities, typos, performance issues, and security issues found by Vidma auditing team. If the code is under development, we 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 deeply investigated the given codebase and analyzed the overall security and performance of smart contracts.

The audit was conducted from March 11th to April 11th, 2022 and the final results are present in this document.

Vidma auditing team has made a review of the following contract:

- BridgeFee;
- BridgeFeeOracle.

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

Initial commit submitted for the audit:
25f5671590385372191 76f042fcfb89140e919fb

Last commit:
9b44ba5f29cb842a46fd382e67feef5ed7ce343b

In order to conduct a more detailed audit, Xend Finance has provided the following documentation:
https://drive.google.com/drive/folders/1ZxqTao1fuq41rnr9BbMDV4wdkxWEWL9V?usp=sharing

WORKFLOW OF THE AUDITING PROCESS

During the manual phase of the audit, Vidma team manually looks through the code in order to find any security issues, typos, or discrepancies with the logic of the contract.

Within the testing part, Vidma auditors run integration tests using the Truffle testing framework. The test coverage and the tests themselves are inserted into this audit report.

Vidma team uses the most sophisticated and contemporary methods and techniques to ensure the contract does not have any vulnerabilities or security risks:

- Re-entrancy;
- Access Management Hierarchy;
- Arithmetic Over/Under Flows;
- Unexpected Ether;
- Delegatecall;
- Default Public Visibility;
- Hidden Malicious Code;
- Entropy Illusion (Lack of Randomness);
- External Contract Referencing;
- Short Address/Parameter Attack;
- Unchecked CALL Return Values;
- Race Conditions / Front Running;
- General Denial Of Service (DOS);
- Uninitialized Storage Pointers;
- Floating Points and Precision;
- Tx.Origin Authentication;
- Signatures Replay;
- Pool Asset Security (backdoors in the underlying ERC-20).

STRUCTURE AND ORGANIZATION OF THE FINDINGS

For the convenience of reviewing the findings in this report, Vidma auditors classified them in accordance with the severity of the issues. (from most critical to least critical). The acceptance criteria are described below.

All issues are marked as "Resolved" or "Unresolved", depending on whether they have been fixed by Xend Finance or not. The latest commit, indicated in this audit report should include all the fixes made.

To ease the explanation, the Vidma team has provided a detailed description of the issues and recommendations on how to fix them.

Hence, according to the statements above, we classified all the findings in the following way:

Finding	Description
Critical	The issue bear a definite risk to the contract, so it may affect the ability to compile or operate.
High	Major security or operational risk found, that may harm the end-user or the overall performance of the contract.
Medium	The issue affects the contract to operate in a way that doesn't significantly hinder its performance.
Low	The found issue has a slight impact on the performance of the contract or its security.
Informational	The issue does not affect the performance or security of the contract/recommendations on the improvements.

MANUAL REPORT

Useless *require* in *setBridgeAmount()* function

High | Resolved

There is zero address check in the *setBridgeAmount()* which is redundant. Min/max values for the zero address of the token needed to be settled in the *setBridgeAmount()* function to transfer the native token via a bridge. In this way, the *transfer(address)* function will not work at all due to the impossibility to set min/max values regardless of the token address.

Recommendation:

Remove *require* statement from *setBridgeAmount()* function for zero token address check.

Incorrect validation in the *setOracleCaller()* function

High | Resolved

In functions *setOracleCaller()* there is a typo in the *require* statement in line 20. Here should be checking if the new input address of the oracle caller *_oracleCaller* isn't a zero address but instead is checked global state variable *oracleCaller*. It makes it impossible to set the *oracleCaller* address so it will always be a zero address.

Recommendation:

Consider changing *oracleCaller* to *_oracleCaller* in line 20.

The contracts are not compiled

High | Resolved

The contracts use incorrect tags when writing NatSpec annotations (@notice is not used for internal and private variables), as well as incorrectly written calls to the call method in the transfer (address) function.

Recommendation:

Solve these problems and compile contracts.

Incorrect precisions in the calculation

High | Invalid

In contract BridgeFee in function *getFeeAmounts()* is incorrect precisions calculation for *feeAmount*.

Steps to reproduce:

_totalAmount: 1 ether;

Fee: value 10, precision 18;

Expected result: *feeAmount* is 0.1 ether, *bridgeAmount* is 0.9 ether ;

Real result: *feeAmount* is 0, *bridgeAmount* is 1 ether.

In this example with fee precision 18, the minimum value that will return the correct fee is 10 ether. So in this case the fee amount will not be charged.

Recommendation:

Check the logic of precisions in the formula in line 77.

Unsafe transfer usage

Medium | Resolved

In contract BridgeFee.sol in function *transfer(address)* is used *transfer()* function for transferring ETH. Since transfers have fixed 2300 gas for executing it is unsafe to use it with new versions of solidity.

Recommendation:

Consider to use *call()* instead of *transfer()*. Don't forget to check the return bool value when will use *call()*.

Unchecked transfer return value

Low | Resolved

In contract BridgeFee, ignored the return value of transferFrom in the function *transfer(address, uint256, address)* in lines51 and 52.

Recommendation:

Consider validating return value by wrapping *transferFrom* in require or using SafeERC20 library.

Lack of zero address check in contract BridgeFee

Low | Resolved

In functions *transfer(address, uint256, address)* and *transfer(address)*, there is no check if the *dcrmAddress* address isn't a zero address.

Recommendation:

Consider adding a check for zero address in functions *transfer(address, uint256, address)* and *transfer(address)*.

Lack of zero value check

Low | Resolved

In contract BridgeFee in function *transfer(address, uint256, address)* there is no check if the *amount* value isn't a zero.

Recommendation:

Consider adding a check for zero value for input parameter *amount* in function *transfer(address, uint256, address)*.

Lack of validation in *setBridgeAmount()* function

Low | Resolved

In contract BridgeFeeOracle in the function *setBridgeAmount()* there is no check if param *max* is bigger than *min*.

Recommendation:

Consider adding an additional validation in the function *setBridgeAmount()*.

Floating pragma

Low | Resolved

The current version of solc in contracts BridgeFee and BridgeFeeOracle 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.

Lack of event emission while the critical data state is changing in contract BridgeFeeOracle

Low | Resolved

In function *setOracleCaller()* critical access control parameter of oracle caller is changed. There is no event emission for these changes to track off-chain change.

Recommendation:

Emit an event for critical parameter changes in function *setOracleCaller()*.

Unspecified state variable visibility

Informational | Resolved

It is best practice to set the visibility of state variables explicitly. The default visibility for *minimumAmounts* and *maximumAmounts* in contract BridgeFeeOracle in lines 9 and 10 is internal.

Recommendation:

Explicitly specify the visibility of state variables.

Lack of NatSpec annotations

Informational | Resolved

Smart contracts BridgeFee and BridgeFeeOracle is not covered by NatSpec annotations.

Recommendation:

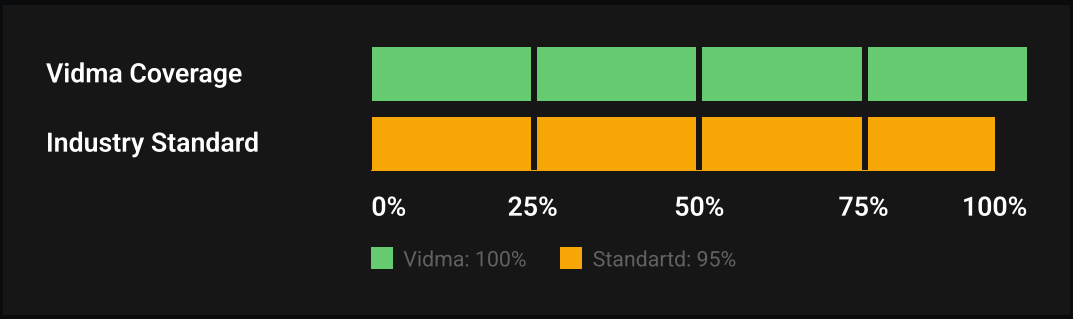
Consider covering by NatSpec all contract methods.

TEST RESULTS

To verify the contract security and performance a bunch of integration tests was made using the Truffle testing framework.

Tests were based on the functionality of the code, business logic, and requirements and for the purpose of finding the vulnerabilities in the contacts.

In this section, we provide tests written by Vidma auditors.



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

Tests are written by Vidma

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	100.00	93.75	100.00	100.00
BridgeFee.sol	100.00	92.31	100.00	100.00
BridgeFeeOracle.sol	100.00	100.00	100.00	100.00
contracts\interfaces\	100.00	100.00	100.00	100.00
IBridgeFee.sol	100.00	100.00	100.00	100.00
IBridgeFeeOracle.sol	100.00	100.00	100.00	100.00
All files	**100.00**	**93.75**	**100.00**	**100.00**

Test Results

```
Contract: BridgeFee
  BridgeFee Set/Get Functions Phase Test Cases
    ✓ should set configure correctly (2022ms)
    ✓ shouldn't set configure if oracle address is zero's
      address (1154ms)
    ✓ shouldn't set configure if fee address is zero's
      address (418ms)
    ✓ shouldn't set configure if caller isn't owner (374ms)
    ✓ should set token fee correctly (738ms)
    ✓ should set default fee correctly (709ms)
  BridgeFee Transfer Functions Phase Test Cases
    ✓ should transfer through token correctly
      [value: 10, precision: 2] (1971ms)
    ✓ should transfer through token correctly
      [value: 10, precision: 3] (1980ms)
    ✓ should transfer through token with small amount correctly
      [value: 10, precision: 4] (1987ms)
    ✓ should transfer through token correctly
      [value: 1, precision: 5] (1867ms)
    ✓ shouldn't transfer amount if dcrmAddress is zero's
      address (281ms)
    ✓ shouldn't transfer amount if amount is zero (255ms)
    ✓ shouldn't transfer amount if bridge failed (1061ms)
    ✓ shouldn't transfer amount if fee transfer failed (1248ms)
    ✓ shouldn't transfer amount if value of min/max
      is unavailable (1026ms)
    ✓ shouldn't transfer amount if limit exceeded (1105ms)
    ✓ should transfer payable correctly (1199ms)
    ✓ shouldn't transfer payable if dcrmAddress is zero's
      address (307ms)
    ✓ shouldn't transfer payable if msg.value is zero (354ms)

Contract: BridgeFeeOracle
  BridgeFeeOracle Set/Get Functions Phase Test Cases
    ✓ should set oracle caller correctly (318ms)
    ✓ shouldn't set oracle caller if oracle caller is zero's
      address (288ms)
    ✓ shouldn't set oracle caller if caller isn't owner (298ms)
    ✓ should update address of the bridge amount correctly (394ms)
```

```
    ✓ should set bridge amount correctly (619ms)
    ✓ shouldn't set bridge amount if max less than min (346ms)
    ✓ shouldn't set oracle caller if caller isn't allowed (512ms)
    ✓ should get bridge amount correctly (639ms)

  27 passing (37s)
```

We are delighted to have a chance to work together with Xend Finance team and contribute to their success by reviewing and certifying the security of the smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

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

VIDMA