



SMART CONTRACT AUDIT

Project: Xend Finance
Date: June 22nd, 2022

TABLE OF CONTENTS

Summary	02
Scope of Work	05
Workflow of the auditing process	06
Structure and organization of the findings	08
Manual Report	10
■ Low Resolved	
Useless function declaration	10
■ Low Resolved	
Lack of event emit while some important data is setted	10
■ Informational Resolved	
Lack of NatSpec annotation	11
■ Informational Resolved	
Naming convention	11
■ Informational Resolved	
Contract name does not match the filename	11
Test Results	12
Tests written by Vidma auditors	13

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.

During the audit process, we reviewed strategy based on Alpaca AUSD protocol to earn profit from depositing funds into the Vault contract. Expect Alpaca AUSD protocol this strategy also relies on Ellipsis Liquidity protocol and Pancakeswap. Audited smart contracts are following security best practices and code style.

During the audit process, the Vidma team found several issues, which were successfully fixed by Xend Finance team. 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	0 issues	0 issues	0 issues
Medium	0 issues	0 issues	0 issues
Low	2 issues	2 issues	0 issues
Informational	3 issues	3 issues	0 issues
Total	5 issues	5 issues	0 issues

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

High Confidence

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



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 June 1st, 2022 to June 22nd, 2022. The outcome is disclosed in this document.

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

- AlpacaAusdEpsStrategy;
- BaseStrategy.

The source code was taken from the following **source**:

<https://github.com/xendfinance/x-vault/tree/alpaca-eps-strategy>

Initial commit submitted for the audit:

[caaad2c9c4ecca7c95914c45b2ce294622127a70](https://github.com/xendfinance/x-vault/commit/caaad2c9c4ecca7c95914c45b2ce294622127a70)

Last commit reviewed by the auditing team:

[67fb80862c67fdc462bd07d7d47256a330c47ff7](https://github.com/xendfinance/x-vault/commit/67fb80862c67fdc462bd07d7d47256a330c47ff7)

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/x-vault/blob/5a9478e966ec1ab5e6df4e076b3afdbc5b964d8b/README.md>

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

Useless function declaration

 Low | Resolved

In BaseStrategy.sol contract there is function `delegatedAssets()` which is return zero. There is no specific implementation of this function in the AlpacaAusdEpsStrategy.sol contract.

Recommendation:

Consider deleting the function or adding specific implementation in the AlpacaAusdEpsStrategy.sol contract to provide actual data.

Lack of event emit while some important data is setted

 Low | Resolved

In the AlpacaAusdEpsStrategy.sol contract should be event emit for state variables overriding `minAlpacaToSell` and `collateralFactor` in the appropriate functions `setMinAlpacaToSell()` and `setCollateralFactor()`.

Recommendation:

Consider the emitting events in these specific cases to be able to track critical data change.

Lack of NatSpec annotation

Informational | Resolved

Some of the functions are covered by the NatSpec annotation. But also there are functions where NatSpec is not provided.

Recommendation:

Consider fully covering SCs with NatSpec annotation.

Naming convention

Informational | Resolved

There is a list of constants in the AlpacaAusdEpsStrategy.sol contract. According to the solidity style guide, the constant should be named with all capital letters with underscores separating words *UPPER_CASE_WITH_UNDERSCORES*.

Recommendation:

Consider changing the naming style of the contents according to the style guide recommendation.

Contract name does not match the filename

Informational | Resolved

Smart contract StrategyAlpacaAUSDEPSFarm doesn't match its filename, AlpacaAusdEpsStrategy.sol. According to the style guide contract, library and interface names should also match their filenames.

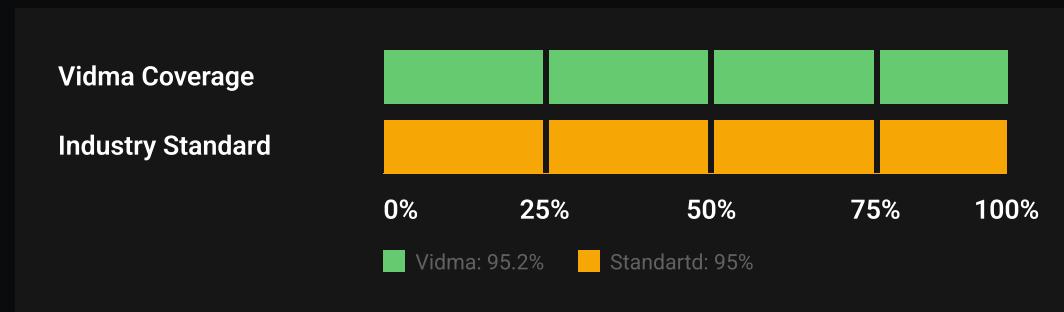
Recommendation:

Consider keeping the same name for the interface and the filename.

TEST RESULTS

To verify the security of contracts 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 industry standards.

Tests written by Vidma auditors

Test Coverage

File	% Stmt	% Branch	% Funcs	% Lines
strategies\	95.98	88.24	100.00	95.63
AlpacaAusdEpsStrategy.sol	95.2	87.5	100.00	94.5
BaseStrategy.sol	96.36	89.81	100.00	97.9
All Files	95.98	88.24	100.00	95.63

Test Results

```
Contract: BaseStrategy
functions:
  harvest
    ✓ shouldn't harvest if caller isn't keeper (788ms)
    ✓ should harvest (5568ms)
    ✓ should harvest if exit is emergency (10541ms)
  harvestTrigger
    ✓ shouldn't harvest strategy is not activated (2252ms)
    ✓ shouldn't harvest if caller isn't keeper (3723ms)
    ✓ should return true when function hadn't been called
      in a while (449ms)
  withdraw
    ✓ shouldn't withdraw if caller is not vault (258ms)
    ✓ should withdraw (619ms)
  migrate
    ✓ shouldn't migrate if caller is not vault or
      governance (186ms)
    ✓ shouldn't migrate if vault address is not the same (234ms)
    ✓ should migrate (797ms)
```

```

apiVersion
  ✓ should return correct version (108ms)
delegatedAssets
  ✓ should return correct delegated assets (97ms)
setStrategist
  ✓ couldn't set with zero address (239ms)
  ✓ should set strategist (269ms)
setKeeper
  ✓ couldn't set with zero address (222ms)
  ✓ should set keeper (294ms)
setRewards
  ✓ shouldn't set with zero address (234ms)
  ✓ should set rewards (299ms)
setProfitFactor
  ✓ should set profit factor (314ms)
setDebtThreshold
  ✓ should set debt threshold (200ms)
isActive
  ✓ should return status of strategy
sweep
  ✓ shouldn't remove tokens if token address is want
    token (285ms)
  ✓ shouldn't remove tokens if token address is vault (221ms)
  ✓ shouldn't remove tokens if token address is protected (471ms)
  ✓ should sweet (1501ms)
setMaxReportDelay
  ✓ should set max report delay (296ms)

```

Contract: StrategyAlpacaAUSDEPSFarm

```

functions:
expectedReturn
  ✓ should return 0 when debt >= estimatedAssets (4822ms)
  ✓ should return sub of estimatedAssets and debt (1852ms)
harvestTrigger
  ✓ should return false when strategy isn't activated (508ms)
  ✓ should return false in the final return (3856ms)
  ✓ should return true when function hadn't been called
    in a while (714ms)
prepareReturn
  ✓ should work when collateral < minAlpacaToSell
  ✓ should work when balance of want is less than needed (312ms)
  ✓ should work when balance of want is >= than needed (207ms)

```

```
name
  ✓ should return correct name (80ms)
setForceMigrate
  ✓ should set flag for forceful migration (625ms)
setMinAlpacaToSell
  ✓ should set minimum amount of alpaca token to sell (284ms)
setDisposalPath
  ✓ should set pancakeswap path (576ms)
setCollateralFactor
  ✓ shouldn't set with zero value (265ms)
  ✓ should set collateral factor (392ms)
```

41 passing (2m)

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.

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

