



# SMART CONTRACT AUDIT

Project: Xend Finance  
Date: February 14th, 2022

# TABLE OF CONTENTS

Summary . . . . .	03
Scope of Work . . . . .	06
Workflow of the auditing process . . . . .	07
Structure and organization of the findings . . . . .	08
Manual Report . . . . .	09
■ Critical   Resolved	
Plt is not possible to make a deposit of tokens, so it is impossible to withdraw them . . . . .	09
■ Medium   Resolved	
Avoid initial values in field declarations . . . . .	09
■ Medium   Resolved	
Function without implementation in AlpacaStrategy contract . . . . .	10
■ Medium   Not valid	
Incorrect condition in <code>prepareMigration()</code> function at AlpacaStrategy contract . . . . .	10
■ Low   Resolved	
Duplicate of the function <code>delegatedAssets()</code> . . . . .	11
■ Low   Resolved	
Duplicate of the function <code>distributeRewards()</code> . . . . .	11
■ Informational   Resolved	
Interface name not match with the filename . . . . .	11
■ Informational   Resolved	
Order of layout . . . . .	12
■ Informational   Resolved	
Empty blocks . . . . .	13

■ Informational   Resolved	
Requires without explanations	sgy.sol . . . . . 13
■ Informational   Resolved	
The condition is not provided for all possible options	. . . . . 13
■ Informational   Resolved	
Unneeded import in the contract BaseStrategy	. . . . . 14
■ Informational   Resolved	
Reassigning a value to a variable	. . . . . 14
■ Informational   Resolved	
Unneeded imports in the contract AlpacaStrategy	. . . . . 14
Test results	. . . . . 15
Tests are written by Vidma	. . . . . 16

# SUMMARY

Vidma team has conducted a smart contract audit for the given codebase.

Contracts are in good condition. Based on the fixes provided by the Xend Finance team and on the quality and security of the codebase provided, the Vidma team can give a score of 97.6 to the audited smart contracts.

During the auditing process, the Vidma team has found 1 critical issue, 2 issues with a medium level of severity, 2 issues with a low level of severity, a couple of informational issues.

Severity of the issue	Total found	Resolved	Unresolved
Critical	1 issue	1 issue	0 issues
High	0 issues	0 issues	0 issues
Medium	2 issues	2 issues	0 issues
Low	2 issues	2 issues	0 issues
Informational	8 issues	8 issues	0 issues
<b>Total</b>	<b>13 issues</b>	<b>13 issues</b>	<b>0 issues</b>

Evaluating the findings, we can assure that the contract is safe to use and all the issues found are performed only by certain conditions and cases. Under the given circumstances we can set the following risk level:

High Confidence

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 run 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 deeply investigated the given codebase and analyzed the overall security and performance of smart contracts.

The debrief took place from Jan 12th to Feb 14th, 2021 and the final results are present in this document.

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

- AlpacaStrategy;
- BaseStrategy;
- IVault;
- IAlpacaFarm;
- IUniswapV2Router;
- VaultAPI.

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

<https://github.com/xendfinance/x-vault/commit/88df2de128b097bc89f3e2480b9c033337d901fc>

**Initial commit** submitted for the audit:

[88df2de128b097bc89f3e2480b9c033337d901fc](https://github.com/xendfinance/x-vault/commit/88df2de128b097bc89f3e2480b9c033337d901fc)

**Last commit:**

[b2d07c2b8e923601c2b92367c3bdacc673ff0331](https://github.com/xendfinance/x-vault/commit/b2d07c2b8e923601c2b92367c3bdacc673ff0331)

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
<span style="color: #C0392B;">■</span> Critical	The issue bear a definite risk to the contract, so it may affect the ability to compile or operate.
<span style="color: #E74C3C;">■</span> High	Major security or operational risk found, that may harm the end-user or the overall performance of the contract.
<span style="color: #F08040;">■</span> Medium	The issue affects the contract to operate in a way that doesn't significantly hinder its performance.
<span style="color: #FDD835;">■</span> Low	The found issue has a slight impact on the performance of the contract or its security.
<span style="color: #2ECC71;">■</span> Informational	The issue does not affect the performance or security of the contract/recommendations on the improvements.

# MANUAL REPORT

**It is not possible to make a deposit of tokens, so it is impossible to withdraw them**

 Critical | Resolved

When testing the `setEmergencyExit()` function, the `withdraw()` function is called from the `alpacaFarm` contract, which checks the caller in the require

```
user.fundedBy == msg.sender, "only funder";
```

To satisfy this condition, it is logical that you first need to call the `deposit()` function from the same contract. However, for correct work, the caller should be a strategy, not just a call from the user directly. The contract does not find a call for this function.

**Recommendation:**

Add the `deposit()` function in the strategy.

## Avoid initial values in field declarations

 Medium | Resolved

As the `AlpacaStrategy` contract should be upgradable by Proxy pattern it is required to not use initial values in field declarations in such cases.

**Recommendation:**

Consider moving initialization of variables `minAlpacaToSell` and `forceMigrate` into the `initialize()` function. Also, as a recommendation, you can avoid directly initializing `forceMigrate` with `false` value as by default bool variables equal `false`.

## Function without implementation in AlpacaStrategy contract

Medium | Resolved

In the AlpacaStrategy contract there is `distributeRewards()` function which has no implementation.

### Recommendation:

Consider adding an implementation to the function or provide any information why AlpacaStrategy contract doesn't need this logic.

## Incorrect condition in `prepareMigration()` function at AlpacaStrategy contract

Medium | Not valid

In the AlpacaStrategy contract in function `prepareMigration()` there is check if migration is allowed. In the current code, there is a possibility to call `prepareMigration()` when the `forceMigrate` variable is equal to `false` instead of `true`. In this case, the `prepareMigration()` function will be able to call immediately after the contract deployment.

### Recommendation:

Consider fixing the condition in the `prepareMigration()` function.

### Re-Audit:

An explanation was provided by the Xend Finance team:

"Generally, `forceMigrate` is false so it forces to withdraw all assets from alpaca protocol and do the migration, but when facing an issue with alpaca protocol so can't withdraw assets, then set `forceMigrate true`, so enables to do the migration without withdrawing assets from alpaca protocol".

## Duplicate of the function `delegatedAssets()`

 Low | Resolved

---

In the contract AlpacaStrategy used the function `delegatedAssets()` that was declared in the contract BaseStrategy without any changes.

### **Recommendation:**

Remove function from AlpacaStrategy contract to follow the DRY pattern.

## Duplicate of the function `distributeRewards()`

 Low | Resolved

---

The implementation for `distributeRewards()` was added to correct a previously found issue regarding the not added implementation of this function. But it is the same as the implementation in BaseStrategy contract so it makes sense to not override `distributeRewards()` in AlpacaStrategy but keep it only in BaseStrategy contract to follow the DRY pattern.

### **Recommendation:**

Remove function from AlpacaStrategy contract to follow the DRY pattern.

## Interface name not match with the filename

 Informational | Resolved

---

Interface `IAlpacaVault` doesn't match with its filename which is `IVault.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.

## Order of layout

 Informational | Resolved

---

The layout contract elements in the BaseStrategy contract are not logically grouped.

Inside each contract, library or interface, use the following order:

- Library declarations (using statements);
- Constant variables;
- Type declarations;
- State variables;
- Events;
- Modifiers;
- Functions.

Functions should be grouped according to their visibility and ordered in the following way:

- Constructor;
- Receive function (if exists);
- Fallback function (if exists);
- External;
- Public;
- Internal;
- Private.

### Recommendation:

Consider changing layout and functions order according to solidity style guide documentation.

## Empty blocks

 Informational | Resolved

---

In both AlpacaStrategy and BaseStrategy contracts there is an empty constructor. As the constructor function is optional you can avoid using it in case of no implementation.

### Recommendation:

Consider removing the empty constructor.

## Requires without explanations

 Informational | Resolved

---

Requires in the functions of the BaseStrategy contract without comments.

### Recommendation:

Add comments to the requirements where needed.

## The condition is not provided for all possible options

 Informational | Resolved

---

The condition in function `expectedReturn()` of the AlpacaStrategy contract is not provided for all possible options, it is better to include all possible options for testing.

### Recommendation:

Change condition (line 92) to if `debt >= estimatedAssets`.

## Unneeded import in the contract BaseStrategy

Informational | Resolved

In the contract, BaseStrategy used unneeded import of interface IERC20.sol because it imports from another interface VaultAPI.sol that was also imported in this contract.

### Recommendation:

Remove IERC20 from import in BaseStrategy contract.

## Reassigning a value to a variable

Informational | Resolved

In the contract BaseStrategy the value of the variable *maxReportDelay* is assigned. The same value is assigned in the contract AlpacaStrategy.

### Recommendation:

Remove reassigning a value to a variable from AlpacaStrategy contract to follow the DRY pattern.

## Unneeded imports in the contract AlpacaStrategy

Informational | Resolved

In the contract AlpacaStrategy used unneeded imports of SafeERC20 for IERC20, SafeMath for uint256, because it is importing in the BaseStrategy contract.

### Recommendation:

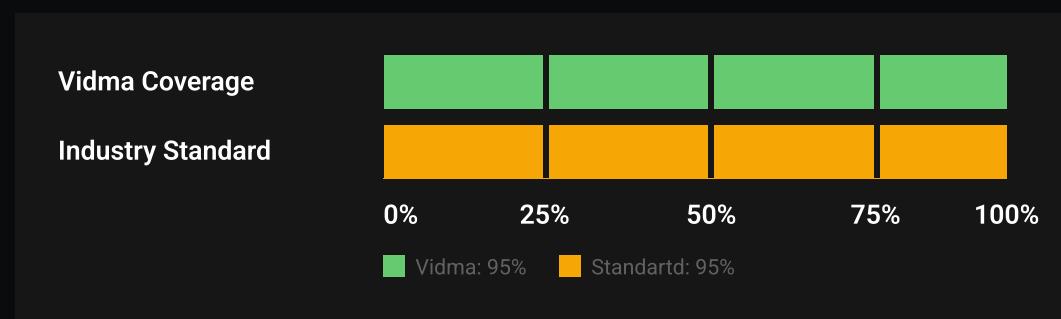
Remove SafeERC20 and SafeMath (lines 11, 13) from AlpacaStrategy contract to follow the DRY pattern.

# TEST RESULTS

To verify the contract security and performance a bunch of integration tests were 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 contracts.

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
interfaces\	100.00	100.00	100.00	100.00
VaultAPI.sol	100.00	100.00	100.00	100.00
interfaces\alpaca\	100.00	100.00	100.00	100.00
IAlpacaFarm.sol	100.00	100.00	100.00	100.00
IAlpacaVault.sol	100.00	100.00	100.00	100.00
interfaces\uniswap\	100.00	100.00	100.00	100.00
IUniswapV2Router.sol	100.00	100.00	100.00	100.00
strategies\	100.00	83.93	100.00	100.00
AlpacaStrategy.sol	95.88	77.38	100.00	95.45
BaseStrategy.sol	94.12	90.48	100.00	95.24
All files	95.00	83.93	100.00	95.34

# Test Results

## Contract: StrategyAlpacaFarm

### StrategyAlpacaFarm Initializing Phase Test Cases

- ✓ should initialize ibToken address correctly (100ms)
- ✓ should initialize maxReportDelay correctly (114ms)
- ✓ should initialize path's addresses correctly (188ms)

### StrategyAlpacaFarm Set/Get Functions Phase Test Cases

- ✓ should get name correctly (128ms)
  - ✓ should set force migrate correctly (623ms)
  - ✓ shouldn't set force migrate if caller isn't governance (486ms)
  - ✓ should set min value to sell correctly (455ms)
  - ✓ shouldn't set min value to sell if caller isn't management (343ms)
  - ✓ should set disposal path correctly (736ms)
  - ✓ should get tend trigger correctly (98ms)
  - ✓ should get expected return correctly (19963ms)
  - ✓ should check price correctly (1628ms)
  - ✓ should do deposit from strategy correctly (1142ms)
- ### StrategyAlpacaFarm Harvest Function Phase Test Cases
- ✓ should get harvest trigger correctly (2934ms)
  - ✓ should get harvest trigger correctly if do harvest several times (3) (19071ms)

## Contract: BaseStrategy

### BaseStrategy Initializing Phase Test Cases

- ✓ should initialize vault address correctly (109ms)
- ✓ should initialize want address correctly (254ms)
- ✓ should approve tokens correctly (126ms)
- ✓ should initialize strategist address correctly (128ms)
- ✓ should initialize rewards address correctly (164ms)
- ✓ should initialize keeper address correctly (123ms)
- ✓ should set ptoected tokens correctly (137ms)
- ✓ should initialize profitFactor correctly (95ms)
- ✓ should initialize debtThreshold correctly (207ms)
- ✓ should initialize maxReportDelay correctly (105ms)
- ✓ shouldn't initialize again (201ms)

### BaseStrategy Set/Get Functions Phase Test Cases

- ✓ should get api version correctly (112ms)
- ✓ should get delegated assets correctly (422ms)

- ✓ shouldn't set strategist if zero's address (276ms)
- ✓ shouldn't set strategist if caller isn't authorized (270ms)
- ✓ should set keeper address correctly (386ms)
- ✓ shouldn't set keeper if zero's address (230ms)
- ✓ should set rewards address correctly (471ms)
- ✓ shouldn't set rewards if zero's address (209ms)
- ✓ shouldn't set rewards if caller isn't strategist (269ms)
- ✓ should get info about active correctly (3538ms)
- ✓ should set profit factor correctly (387ms)
- ✓ should set debt threshold correctly (363ms)
- ✓ should set report delay correctly (329ms)

#### BaseStrategy Withdraw Functions Phase Test Cases

- ✓ should withdraw alien tokens from this strategy correctly (1429ms)
- ✓ should get harvest trigger correctly if do harvest several times (4) (19071ms)
- ✓ shouldn't withdraw alien tokens from this strategy if token is want's address (268ms)
- ✓ shouldn't withdraw alien tokens from this strategy if token is vault's address (300ms)
- ✓ shouldn't withdraw alien tokens from this strategy if token is protected (392ms)
- ✓ shouldn't withdraw assets to the vault if caller isn't vault (303ms)

#### BaseStrategy Harvest & Emergency Exit Functions Phase Test Cases

- ✓ should do harvest correctly if emergencyExit is false (2169ms)
- ✓ should do harvest correctly if emergencyExit is true (6059ms)
- ✓ shouldn't do harvest if caller isn't keeper, strategist or governance (340ms)
- ✓ should get harvest trigger correctly (16151ms)
- ✓ should do harvest twice correctly (8955ms)
- ✓ should set emergency exit correctly (4734ms)
- ✓ shouldn't set emergency exit if can't be sent balance of strategy from alpaca token (6644ms)

#### BaseStrategy Migrate Function Phase Test Cases

- ✓ should migrate strategies correctly (16971ms)
- ✓ shouldn't migrate strategy if caller isn't vault or governance (2356ms)



```
✓ shouldn't migrate strategy if new strategy isn't
  vault (24764ms)
BaseStrategy Tend Function Phase Test Cases
✓ should do tend correctly (2335ms)
```

57 passing (4m)



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.