IPOR

SMART CONTRACT AUDIT

zokyo

October 24th 2022 | v. 1.1

Security Audit Score

PASS

Zokyo Security has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

TECHNICAL SUMMARY

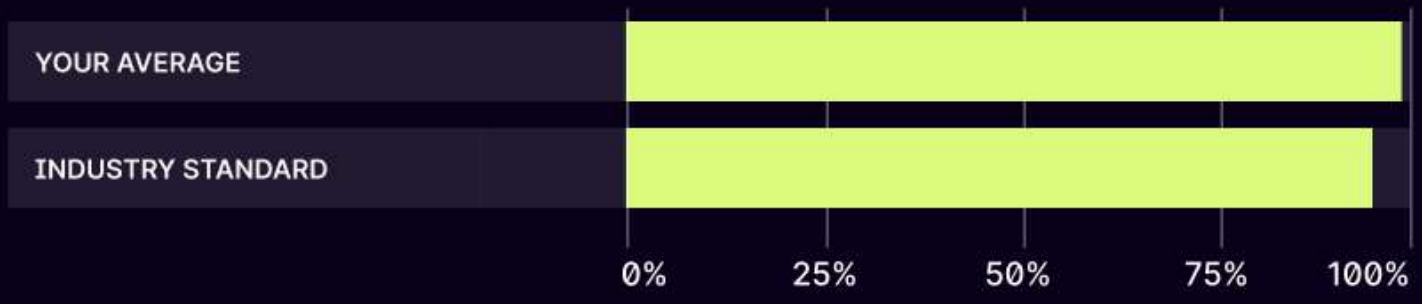
This document outlines the overall security of the IPOR smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the IPOR smart contract codebase for quality, security, and correctness.

Contract Status

There were no critical issues found during the audit. (See Complete Analysis)
Testable Code (please check the example below)

Testable Code



The testable code is 99.49%, which is above the industry standard of 95%. (See Complete Analysis)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the IPOR team put in place a bug bounty program to encourage further and active analysis of the smart contract.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the IPOR repository.

Repository: https://github.com/IPOR-Labs/ipor-protocol/commit/2a7cf870657f6ea0f7a356df79d7a60d5a3d2_713

Last commit: b12ef5b7f2f2dab26e2a787ada5d3c895dcbd83e

Within the scope of this audit, Zokyo auditors have reviewed the following contract(s):

- Milton.sol
- MiltonDai.sol
- MiltonInternal.sol
- MiltonStorage.sol
- MiltonUsdc.sol
- MiltonUsdt.sol
- IporSwapLogic.sol
- SoapIndicatorLogic.sol
- Joseph.sol
- JosephDai.sol
- JosephInternal.sol
- JosephUsdc.sol
- JosephUsdt.sol
- AmmMiltonStorageTypes.sol
- AmmMiltonTypes.sol
- MiltonSpreadInternal.sol
- MiltonSpreadModel.sol
- MiltonSpreadModelDai.sol
- MiltonSpreadModelUsdc.sol
- MiltonSpreadModelUsdt.sol
- IporOracleFacadeDataProvider.sol
- MiltonFacadeDataProvider.sol
- CockpitDataProvider.sol
- IporErrors.sol
- IporOracleErrors.sol
- JosephErrors.sol
- MiltonErrors.sol
- StanleyErrors.sol
- Constants.sol
- PaginationUtils.sol
- IporMath.sol
- IporOracle.sol
- IpToken.sol
- IvToken.sol
- IporOwnable.sol
- IporOwnableUpgradeable.sol
- Stanley.sol
- StanleyDai.sol
- StanleyUsdc.sol
- StanleyUsdt.sol
- IpToken.sol
- IvToken.sol
- StrategyAave.sol
- StrategyCompound.sol
- StrategyCore.sol
- MiltonUsdt

Throughout the review process, Zokyo Security ensures that the contract:

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of resources, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of IPOR smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01	Due diligence in assessing the overall code quality of the codebase.	02	Cross-comparison with other, similar smart contracts by industry leaders.
03	Testing contract logic against common and uncommon attack vectors.	04	Thorough manual review of the codebase, line by line.

Executive Summary

There were no critical issues found during the audit. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner.

Contracts are well written and structured. The findings during the audit have no impact on contract performance or security, so it is fully production-ready.

Despite the fact, the expected logic is managing all vestings by the owner, it should be careful with parameters to avoid mistakes during the vesting process.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For easier navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Customer’s side or is an issue that the Customer disregards as a problem. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that it can lead to a significant loss, funds may be lost or allocated incorrectly.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

SYSTEM OVERVIEW

The core components of the IPOR Protocol are the Ipor Index, Liquidity Pools, AMM and IPOR Oracle. During the initial assessment of the protocol it has been discovered that at the center of these components is the price oracle and the price feeds it exposes. In most DeFi protocols, Oracles play an important role in securing the funds deposited within liquidity pools. After analyzing the oracle usage throughout the project, it has been established that further clarifications should be made. As such, Zokyo team engaged with the IPOR team over video call to discuss this component and the sources data is fed from.

It has been concluded that the smart contract implementation and how it's used throughout the components that consume feeds is correct and meets the set goals. After engagement with the IPOR team a conclusion has been reached regarding the Oracle long-term design and implementation. It has also been concluded that the current form of implementation, both on-chain and off-chain, is sufficient for the requirements specified by the IPOR team. Different proposals have been made regarding the centralized nature of the oracle and how feeds could be disrupted due to external/off-chain technical issues. It has been concluded that current existing oracle solutions, such as ChainLink, don't fully meet the requirements regarding composability and gas costs. So the decision has been made by the IPOR team to maintain the current solution and after ensuring a stable deployment and setup of the protocol, to gradually transition to a more decentralized approach. IPOR presented their design for the oracle following a review of resources requested from the Zokyo team. Although not in the scope of this report, support has been provided.

The IPOR protocol is a complex set of contracts and as such, to mitigate possible issues in the future or the need to accommodate new features, all contracts use the Oppenzeppelin's implementation of Universal Upgradeable Proxy Pattern (UUPS EIP-1822) for upgradeability. This is correctly implemented but no plans for governance have been communicated to the Zokyo team.

During the audit process the IPOR team found issues/optimizations which were discussed and fixed in pull requests reviewed by Zokyo. Some of the issues/optimizations were found by Zokyo and forwarded. These also got fixed and reviewed so are not part of this report.

COMPLETE ANALYSIS

FINDINGS SUMMARY

№	TITLE	RISK
1	Multiple external calls are executed in the same transaction (MiltonInternal)	Low
2	Multiple external calls are executed in the same transaction (StrategyAave)	Low
3	Incorrect balance assertion (MiltonInternal)	Informational
4	Incorrect balance assertion (Joseph)	Informational
5	Redundant usage of SafeMath (IvToken)	Low
6	Variable shadowing	Informational
7	Logical operator gas optimization	Informational

COMPLETE ANALYSIS

LOW | UNRESOLVED

Multiple external calls are executed in same transaction
In contract MiltonInternal, at line 348, there's an external call performed after another external call that queries the IPOR price oracle for the accruedIbtPrice. There are no checks for the returned value of the oracle, which can lead to undefined behavior if the oracle price feeds were not updated or the feeds are corrupted. In the current implementation, with the codebase of the Oracle being maintained by the IPOR team, this is not a problem. But given the oracle only defines an interface, and following discussions for the long-term goals of the oracle, this might change and the price feeds source might not be under the team's control.

Recommendation:
Refactor the flow such as the values returned by the oracle are taking into consideration the potential vulnerabilities mentioned above.

LOW | RESOLVED

Multiple external calls are executed in the same transaction
In contract StrategyAave at line 74, there's an external call performed that depends on another external call result. The first call retrieves the address of the lending pool for a certain asset and then queries the lending pool for its reserves data. Given the LendingPoolAddressProvider is a contract deployed by a 3rd party this could return invalid data and result in undefined behavior.

Recommendation:
Add a require check for zero address before querying the Aave lending pool to prevent undefined behavior.

INFORMATIONAL | ACKNOWLEDGED

Incorrect balance assertion

In contract MiltonInternal at line 328, in function _getAccruedBalance the liquidityPool balance is checked before returning the accruedBalance. The statement checks for greater than or equal to 0 balances, which is incorrect because for a 0 balance it would not revert.

```
require(liquidityPool >= 0, MiltonErrors.LIQUIDITY_POOL_AMOUNT_TOO_LOW);
accruedBalance.liquidityPool = liquidityPool.toUint256();
```

Recommendation:

Check the result returned by the function for the case when liquidityPool is 0 to avoid getting undefined behavior because of accruedBalance being 0.

INFORMATIONAL | ACKNOWLEDGED

Incorrect balance assertion

In contract Joseph at line 72, in function _calculateExchangeRate the balance is checked before calculating and returning the exchange rate. The statement checks for greater than or equal to 0 balances, which is incorrect because for a 0 balance it would not revert.

```
int256 balance = milton.getAccruedBalance().liquidityPool.toInt256() - soap;
require(balance >= 0, MiltonErrors.SOAP_AND_LP_BALANCE_SUM_IS_TOO_LOW);
```

Recommendation:

Check the result returned by the function for the case when balance is 0 to avoid getting undefined behavior because of exchangeRate being 0.

LOW | RESOLVED

Redundant usage of SafeMath

In contract IvToken at line 13, there's a using statement for uin256 involving SafeMath. Given all the contracts use Solidity 0.8.14 the SafeMath library is redundant, as in Solidity >=8.0.0 there can be no under/overflow.

Recommendation:

Remove the using statement for uin256 and the SafeMath library import.

INFORMATIONAL | RESOLVED

Variable shadowing

In contract MiltonStorage there are multiple instances of variable shadowing. It occurs in functions with named return types which are then redeclared inside the function body. It is present at lines 182, 203 and 235

Recommendation:

Remove duplicate declarations

Logical operator gas optimization

Milton → 208, 222, 280, 379, 500, 672, 701, 796, 814

MiltonStorage → 172, 193, 219, 295, 419, 432, 448, 487, 766, 788

SoapIndicatorLogic → 80

Joseph → 76, 85, 99, 123, 130, 187

JosephInternal → 68, 120

MiltonSpreadModel → 58, 76

MiltonFacadeProvider → 116

IporMath → 33

IporOracle → 77, 95, 179, 210

lpToken → 52, 58

lvToken → 48, 54

Stanley → 85, 120, 259, 452

StrategyCompund → 135

In contracts above, for comparison between unsigned integers and the value `0` the “!=” operator was used. Based on the fact that all the variables are unsigned integers, comparison could be done with the ">” operator with the same result. Using the “>” operator the gas usage is 6x cheaper for this particular operation.

Recommendation:

Replace all comparison operators “!=” with “>”, as is described above.

COMPLETE ANALYSIS

AUTOMATED TESTING

Automated testing techniques can improve the manual security review by providing insights into variables dependencies, how functions behave under fuzzing etc. Although helpful, automated testing alone is not enough, as it has its limitations. During the audit we used Mythril and Slither to analyze the smart contracts in depth.

SLITHER

Slither was used for detection of vulnerable solidity code and also for code optimization. During the analysis, no high/critical errors were found, most of them being even false positives.
The table below shows the files where findings were found. The files which are not mentioned, having no findings or only false-positives findings.

Contract	Findings	Description	Security
JosephInternal	JosephInternal._asset (contracts/amm/pool/JosephInternal.sol#35) is never initialized	The variable _asset is used in the contract JosephInternal without being initialized	False-positive
	JosephInternal._ipToken (contracts/amm/pool/JosephInternal.sol#36) is never initialized	The variable _ipToken is used in the contract JosephInternal without being initialized	False-positive
	JosephInternal._milton (contracts/amm/pool/JosephInternal.sol#37) is never initialized	The variable _milton is used in the contract JosephInternal without being initialized	False-positive
	JosephInternal._miltonStorage (contracts/amm/pool/JosephInternal.sol#38) is never initialized	The variable _miltonStorage is used in the contract JosephInternal without being initialized	False-positive
	JosephInternal._stanly (contracts/amm/pool/JosephInternal.sol#39) is never initialized	The variable _stanly is used in the contract JosephInternal without being initialized	False-positive

COMPLETE ANALYSIS

Contract	Findings	Description	Security
MiltonStorage	MiltonStorage._updateSoapIndicatorsWhenOpenSwapPayFixed(uint256,uint256,uint256,uint256) (contracts/amm/MiltonStorage.sol#810-833) ignores return value by rebalanceWhenOpenSwap(openTimestamp,notional,fixedInterestRate,ibtQuantity) (contracts/amm/MiltonStorage.sol#825)	In the contract MiltonStorage the external function rebalanceWhenOpenSwap is called without taking into account the value it returns	False-positive
MiltonFacadeDataProvider	MiltonFacadeDataProvider.getMySwaps(address,uint256,uint256).totalCount_scope_0 (contracts/facades/MiltonFacadeDataProvider.sol#122) is a local variable never initialized	In the contract MiltonFacadeDataProvier exists shadowed and unused variables	Low

Following the automatic analysis in contract JosephInternal, the variables_asset, _ipToken, _milton, _miltonStorage, _stanley are used throughout the contract without being initialized, but taking into account that the purpose of the contract is to be used by Joseph, and the initialization is done there, the result generated by Slither is false-positive. However, that could be a problem, if JosephInternal will be used with another implementation of Joseph which will not initialize the variables.
Following the automatic analysis with slither, a warning occurred because the value returned by the external function: rebalanceWhenOpenSwap was not used. After manual verification of the code, it was found that the result of the function was not needed, which turns the result offered by Slither into a false-positive one.
After the analysis done with Slither, it turned out that in the contract MiltonFacadeDataProvider, function getMySwaps, the variable totalCount is shadowed in the function body and the variable swaps is declared but not used.

COMPLETE ANALYSIS

MYTHRIL

Mythril was used as a standalone tool and in conjunction with Scribble, especially to test the ERC-20 tokens properties. There were no high/critical issues found during this analysis and most of the low/medium findings were discarded as false positives.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Security

As part of our work assisting IPOR in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the IPOR contract requirements for details about issuance amounts and how the system handles these.

IpToken
- Deploy
  - ✓ with wrong asset address (383ms)
- decimals
  - ✓ decimals should be 18 (118ms)
- getAsset
  - ✓ asset should be same with constructor param (45ms)
- setJoseph
  - ✓ set with success (57ms)
  - ✓ wrong address (82ms)
- mint
  - ✓ mint with success (148ms)
  - ✓ wrong amount (130ms)
  - ✓ if caller not Joseph should revert (95ms)
- burn
  - ✓ burn with success (151ms)
  - ✓ wrong amount (90ms)
  - ✓ if caller not Joseph should revert (69ms)
- tokens
  - ✓ deploy IpTokenUsdt (3618ms)
  - ✓ deploy IpTokenUsdc (3431ms)
  - ✓ deploy IpTokenDai (3984ms)

IporOracle
- initialize
  - ✓ init with success (1077ms)
  - ✓ wrong asset address (289ms)
- getVersion
  - ✓ version should be 2 (578ms)
  - ✓ version should greater than 0 (627ms)
- getIndex
  - ✓ return with success (585ms)

- ✓ asset not supported (526ms)

getAccruedIndex
- ✓ return with success (549ms)
- ✓ asset not supported (534ms)

calculateAccruedIbtPrice
- ✓ return with success (456ms)

Authorize Upgrades
- ✓ should allow only owner to execute (544ms)

updateIndex
- ✓ return with success (561ms)
- ✓ call not from updater should revert (553ms)

updateIndexses
- ✓ return with success (654ms)
- ✓ diff length for assets and indexes (501ms)
- ✓ update for unexisting asset should fail (508ms)
- ✓ astUpdateTimestamp > updateTimestamp (536ms)

addUpdater
- ✓ add Updater with success (494ms)
- ✓ call not from owner (397ms)

removeUpdater
- ✓ remove Updater with success (407ms)
- ✓ call not from owner (542ms)

isUpdater
- ✓ updater exists (538ms)
- ✓ updater not exists (459ms)

addAsset
- ✓ add asset with success (579ms)
- ✓ add an asset with wrong address should revert (605ms)
- ✓ add an existing assets should revert (570ms)

removeAsset
- ✓ remove asset with success (611ms)
- ✓ remove an asset with wrong address should revert (690ms)
- ✓ remove a non existing assets should return (733ms)

pause
- ✓ set pause true (616ms)
- ✓ set pause false (605ms)

IporLogic

accrueQuasiIbtPrice
- ✓ accrueTimestamp < indexTimestamp should revert

calculateExponentialWeightedMovingVariance
- ✓ alpha too high should revert

- ✓ indexValue > exponentialMovingAverage
- ✓ spread cant be higher than 1 (38ms)

DecayFactorCalculation

calculate

- ✓ interval one
- ✓ interval two
- ✓ interval three

IvToken

Deploy

- ✓ with wrong asset address (219ms)

decimals

- ✓ decimals should be 18 (54ms)

getAsset

- ✓ asset should be same with constructor param (52ms)

setStanley

- ✓ set with success (78ms)
- ✓ wrong address (74ms)

mint

- ✓ mint with success (99ms)
- ✓ wrong amount (117ms)
- ✓ if caller not Joseph should revert (136ms)

burn

- ✓ burn with success (127ms)
- ✓ wrong amount (125ms)
- ✓ if caller not Joseph should revert (83ms)

tokens

- ✓ deploy IvTokenUsdt (3419ms)
- ✓ deploy IvTokenUsdc (4184ms)
- ✓ deploy IvTokenDai (3823ms)

Stanley

- ✓ try initialize all fails (1400ms)
- ✓ should check total balance (968ms)
- ✓ should call authorize upgrade as not the owner (1271ms)
- ✓ should check calculate exchange rate (1093ms)
- ✓ should set milton (1198ms)
- ✓ should check version (1569ms)
- ✓ should check asset (1584ms)
- ✓ should deposit (2206ms)
- ✓ should check pause/unpause (1981ms)

nice2, youe

- ✓ should withdraw, asset balance aave bigger then compound and amount is equal (2964ms)
- ✓ should withdraw, asset balance aave lesseer then compound (2776ms)

- ✓ should withdraw all, all checks (2096ms)
- ✓ should withdraw all, with no aave strategy balance (2376ms)
- ✓ should withdraw all, with no compound strategy balance (2423ms)
- ✓ should migrate assets to strategy, all checks (2742ms)
- ✓ should set strategy aave with already strategy in store slot (2683ms)
- ✓ should set strategy compound with already strategy in store slot (4088ms)
- ✓ should call authorize upgrade as not the owner (2438ms)
- ✓ should calculate exchange rate with total balance bigger then 0 iv token balance bigger then 0 (270ℓ
- ✓ should withdraw from strategy with transfer set as false (2122ms)
- ✓ should select strategy and withdraw amount (2209ms)

StanleyUsdc

- ✓ try initialize all fails (2002ms)
- ✓ should check total balance (2627ms)
- ✓ should check calculate exchange rate (2246ms)
- ✓ should set milton (2104ms)
- ✓ should get milton (2028ms)
- ✓ should check version (1786ms)
- ✓ should check asset (1787ms)
- ✓ should check pause/unpause (1854ms)
- ✓ should withdraw, asset balance aave bigger then compound and amount is equal (3147ms)
- ✓ should withdraw, asset balance aave lesseer then compound (5780ms)
- ✓ should migrate assets to strategy, all checks (2198ms)
- ✓ should set strategy aave with already strategy in store slot (2257ms)
- ✓ should set strategy compound with already strategy in store slot (2370ms)
- ✓ should call authorize upgrade as not the owner (1813ms)
- ✓ should calculate exchange rate with total balance bigger then 0 iv token balance bigger then 0 (1872ms)
- ✓ should withdraw from strategy with transfer set as false (1893ms)

StanleyUsdt

- ✓ try initialize all fails (1543ms)
- ✓ should check total balance (1984ms)
- ✓ should check calculate exchange rate (2062ms)
- ✓ should set milton (2005ms)
- ✓ should check version
- ✓ should check asset (2248ms)
- ✓ should check pause/unpause (2207ms)
- ✓ should withdraw, asset balance aave bigger then compound and amount is equal (3493ms)
- ✓ should withdraw, asset balance aave lesseer then compound (2973ms)
- ✓ should migrate assets to strategy, all checks (2958ms)
- ✓ should set strategy aave with already strategy in store slot (2907ms)
- ✓ should set strategy compound with already strategy in store slot (4719ms)
- ✓ should call authorize upgrade as not the owner (2335ms)

- ✓ should calculate exchange rate with total balance bigger then 0 iv token balance bigger then 0 (2518ms)
- ✓ should withdraw from strategy with transfer set as false (2532ms)

Core strategy

- ✓ should be able to use core strategy utilities (837ms)

Aave strategy

- ✓ should revert for zero input addresses (892ms)
- ✓ should get apr from reserve data (351ms)
- ✓ should get balance of sharetoken (90ms)
- ✓ should set stkAave or revert for invalid address (138ms)
- ✓ should execute beforeClaim or revert if treasury not set (244ms)
- ✓ should deposit to lending pool (526ms)
- ✓ should withdraw from lending pool (422ms)
- ✓ should execute doClaim (605ms)
- ✓ should revert claim if treasury not set (62ms)

Compound strategy

- ✓ should revert deploy for zero addresses (497ms)
- ✓ should get apr (245ms)
- ✓ should get balance of share token (283ms)
- ✓ should deposit asset (381ms)
- ✓ should withdraw asset (625ms)
- ✓ should execute doClaim or revert (1028ms)
- ✓ should set blocks per year or revert (133ms)

Coverage test

- ✓ Should test coverage for IporOracleFacadeDataProvider (578ms)
- ✓ Should test coverage for CockpitDataProvider (1565ms)
- ✓ Should test coverage for MiltonFacadeDataProvider (1583ms)

Joseph

- ✓ Should be able to initialize properly (4669ms)
- ✓ Should be able to initialize in a paused state (591ms)
- ✓ Should be able to check vault reserves ratio (787ms)
- ✓ Should be able to calculate exchange rate (741ms)
- ✓ Should be able to calculate Redeemed Utilization Rate (656ms)
- ✓ Should be able to provide liquidity (828ms)
- ✓ Should be able to redeem (1692ms)

JosephDai

- ✓ Should be able to initialize properly (4810ms)
- ✓ Should be able to initialize in a paused state (601ms)
- ✓ Should be able to check vault reserves ratio (832ms)
- ✓ Should be able to calculate exchange rate (793ms)
- ✓ Should be able to calculate Redeemed Utilization Rate (595ms)
- ✓ Should be able to provide liquidity (851ms)
- ✓ Should be able to redeem (1934ms)

JosephUsdt

- ✓ Should be able to initialize properly (4917ms)
- ✓ Should be able to initialize in a paused state (665ms)
- ✓ Should be able to check vault reserves ratio (889ms)
- ✓ Should be able to calculate exchange rate (848ms)
- ✓ Should be able to calculate Redeemed Utilization Rate (639ms)
- ✓ Should be able to provide liquidity (934ms)
- ✓ Should be able to redeem (1881ms)

JosephInternal

- ✓ Should be able to get version (62ms)
- ✓ Should be able to set asset address (146ms)
- ✓ Should set rebalance ratio (332ms)
- ✓ Should be able to get redeem fee rate (67ms)
- ✓ Should be able to get Redeem Lp Max Utilization Rate (82ms)
- ✓ Should be able to set treasury manager (178ms)
- ✓ Should be able to get treasury manager (168ms)
- ✓ Should be able to set charlie treasury manager (196ms)
- ✓ Should be able to get charlie treasury manager (174ms)
- ✓ Should be able to set charlie treasury (193ms)
- ✓ Should be able to get charlie treasury (171ms)
- ✓ Should be able to pause contract (189ms)
- ✓ Should be able to unpause contract (372ms)
- ✓ Should be able to set treasury (209ms)
- ✓ Should be able to get treasury (180ms)
- ✓ Should deposit to stanley (244ms)
- ✓ Should be able to transfer to treasury (1454ms)
- ✓ Should be able to transfer to charlie treasury (999ms)
- ✓ Should be able to withdraw from stanley (302ms)
- ✓ Should be able to withdraw all from stanley (374ms)
- ✓ Should be able to rebalance (1332ms)
- ✓ Should be able to set max lp account contribution (141ms)
- ✓ Should be able to get max lp account contribution (421ms)
- ✓ Should be able to set max liquidity pool balance (159ms)
- ✓ Should be able to get max liquidity pool balance (414ms)
- ✓ Should be able to authorize Upgrade (134ms)

Milton

- ✓ Should be initialized properly (5147ms)
- ✓ Should be able to initialize contract in paused state (864ms)
- ✓ Should be able to directly call transfer of derivative amount (901ms)
- ✓ Should be able to calculate spread (1221ms)
- ✓ Should be able to calculate soap (1014ms)
- ✓ Should be able to calculate split opening fee amount (1031ms)

- ✓ Should be able to open Swap Pay Fixed (3885ms)
- ✓ Should be able to calculate Income Fee Value (704ms)
- ✓ Should be able to calculate swap indicators (811ms)
- ✓ Should be able to validate liquidity pool utylization (1268ms)
- ✓ Should be able to transfer tokens based on payoff (1524ms)
- ✓ Should be able to close inactive swaps (989ms)
- ✓ Should be able to close swaps (1493ms)
- ✓ Should be able to close swap pay fixed (1699ms)
- ✓ Should be able to emergency Close fixed swap pay only when paused (1648ms)
- ✓ Should be able to close fixed swap received (1535ms)
- ✓ Should be able to emergency Close fixed swap received when contract is paused (1553ms)
- ✓ Should be able to emergency Close fixed swaps received when contract is paused (1770ms)
- ✓ Should be able to emergency Close swaps fixed pay received when contract is paused (1809ms)
- ✓ Should be able to open swap receive fixed (3469ms)
- ✓ Should be able to authorize Upgrade (788ms)

MiltonUsdt

- ✓ Should be initialized properly (5864ms)
- ✓ Should be able to initialize contract in paused state (931ms)
- ✓ Should be able to calculate spread (1355ms)
- ✓ Should be able to calculate soap (1251ms)
- ✓ Should be able to calculate split opening fee amount (1128ms)
- ✓ Should be able to open Swap Pay Fixed (3662ms)
- ✓ Should be able to calculate Income Fee Value (1084ms)
- ✓ Should be able to calculate swap indicators (976ms)
- ✓ Should be able to validate liquidity pool utylization (924ms)
- ✓ Should be able to transfer tokens based on payoff (1621ms)
- ✓ Should be able to close inactive swaps (1248ms)
- ✓ Should be able to close swaps (2087ms)
- ✓ Should be able to close swap pay fixed (2375ms)
- ✓ Should be able to emergency Close fixed swap pay only when paused (1994ms)
- ✓ Should be able to close fixed swap received (1847ms)
- ✓ Should be able to emergency Close fixed swap received when contract is paused (1618ms)
- ✓ Should be able to emergency Close fixed swaps received when contract is paused (2184ms)
- ✓ Should be able to emergency Close swaps fixed pay received when contract is paused (2011ms)
- ✓ Should be able to open swap receive fixed (4506ms)

MiltonInternal

- ✓ Should be able to get version (5757ms)
- ✓ Should be able to get asset (1299ms)
- ✓ Should be able to get Ipor PublicationFee (1142ms)
- ✓ Should be able to get income Fee rate (914ms)
- ✓ Should be able to get opening Fee rate (1191ms)
- ✓ Should be able to get Opening Fee Treasury Portion Rate (1050ms)

- ✓ Should be able to get max leverage (1173ms)
- ✓ Should be able to get min leverage (1085ms)
- ✓ Should be able to get liquidation deposit amount (996ms)
- ✓ Should be able to get wad liquidation deposit amount (997ms)
- ✓ Should be able to get max swap collateral amount (1006ms)
- ✓ Should be able to get max LP utilization rate (1103ms)
- ✓ Should be able to get max LP utilization per leg rate (1159ms)
- ✓ Should be able to pause contract (1272ms)
- ✓ Should be able to unpause contract (1568ms)
- ✓ Should be able to set joseph (1230ms)
- ✓ Should be able to get joseph (1307ms)
- ✓ Should be able to set up max allowance for asset (1395ms)
- ✓ Should be able to get milton spread model (978ms)
- ✓ Should be able to set spread model (1319ms)
- ✓ Should be able to get accrued balance (1929ms)
- ✓ Should be able to deposit to stanley (2099ms)
- ✓ Should be able to withdraw from stanley (1788ms)
- ✓ Should be able to withdrawal from stanley (1929ms)
- ✓ Should be able to get calculatePayoffPayFixed (1580ms)
- ✓ Should be able to get calculatePayoffReceiveFixed (1509ms)
- ✓ Should be able to get calculateSoapAtTimestamp (1593ms)

MiltonStorage

- ✓ Should be initialized correctly (221ms)
- ✓ Should be able to get version (227ms)
- ✓ Should be able to set joseph (798ms)
- ✓ Should be able to add liquidity (1533ms)
- ✓ Should be able to remove liquidity (1082ms)
- ✓ Should be be able to get liquidity pool account contribution (591ms)
- ✓ Should be able to pause contract (377ms)
- ✓ Should be able to unpause contract (554ms)
- ✓ Should be able to set milton (495ms)
- ✓ Should be able to get extended balance (265ms)
- ✓ Should be able to update storage when transfer to treasury (1269ms)
- ✓ Should be able to update storage when transfer to charlie treasury (1734ms)
- ✓ Should be able to update storage when deposit to stanley (1744ms)
- ✓ Should be able to update storage when withdraw from stanley (1520ms)
- ✓ Should be able to update storage when open swap pay fixed (831ms)
- ✓ Should be able to update storage when open swap receive fixed (622ms)
- ✓ Should be able to get last swap id (647ms)
- ✓ Should be able to get total outstanding notional (349ms)
- ✓ Should be able to get swap pay fixed (386ms)
- ✓ Should be able to get swap receive fixed (437ms)

```
    ✓ Should be able to calculate soap (240ms)
    ✓ Should be able to get swaps fixed (466ms)
    ✓ Should be able to get swaps receive fixed (900ms)
    ✓ Should be able to get swap pay fixed ids (973ms)
    ✓ Should be able to get swaps receive fixed (971ms)
    ✓ Should be able to calculate soap (249ms)
    ✓ Should be able to calculate soap pay fixed (208ms)
    ✓ Should be able to update balance when close swap (712ms)
    ✓ Should be able to update storage when close swap pay fixed (2819ms)
    ✓ Should be able to update storage when close swap receive fixed (3700ms)
    ✓ Should be able to get Swap Ids (1050ms)
    ✓ Should be able to authorize Upgrade (197ms)
  301 passing (13m)
```

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	% Uncovered Lines
amm/	99.78	97.33	98.56	99.79	
Milton.sol	100	97.3	97.14	100	
MiltonDai.sol	100	100	100	100	
MiltonInternal.sol	100	100	100	100	
MiltonStorage.sol	99.56	97.06	98.08	99.56	245
MiltonUsdc.sol	100	100	100	100	
MiltonUsdt.sol	100	100	100	100	
amm/pool/	100	98.21	98	100	
Joseph.sol	100	93.75	100	100	
JosephDAI.sol	100	100	100	100	
JosephInternal	100	100	97.37	100	
JosephUsdc.sol	100	100	100	100	
JosephUsdt.sol	100	100	100	100	
oracles/	100	100	95.24	100	

IporOracle.sol	100	100	95.24	100	
tokens/	100	100	100	100	
IpToken	100	100	100	100	
IvToken	100	100	100	100	
vault/	99.34	96.67	96.55	99.34	
Stanley	99.33	96.67	96.15	99.33	206
StanleyDai	100	100	100	100	
StanleyUsdc	100	100	100	100	
StanleyUsdt	100	100	100	100	
vault/strategies/	97.84	100	96.77	97.87	
StrategyAave	95.89	100	100	95.59	155,156,157
StrategyCompound	100	100	100	100	
StrategyCore	100	100	92.86	100	
All files	**99.49**	**98.06**	**97.93**	**99.49**	

We are grateful to have been given the opportunity to work with the IPOR team.

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

Zokyo's Security Team recommends that the IPOR team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo