SMART CONTRACT AUDIT

ZOKYO.

Jan 11th, 2022 | v. 1.0

PASS

Zokyo Security team has concluded that these smart contracts pass security qualifications and are fully production-ready

SCORE

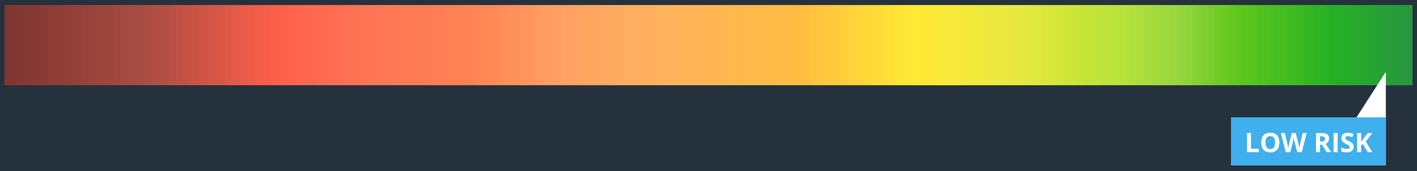
97

TECHNICAL SUMMARY

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

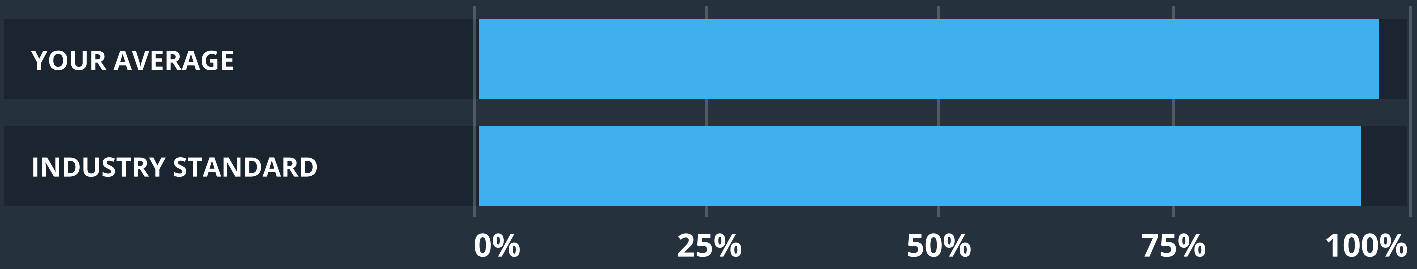
The scope of this audit was to analyze and document the Saffron smart contract codebase for quality, security, and correctness.

Contract Status



There were no critical issues found during the audit.

Testable Code



The testable code is 97%, which is above the industry standard of 95%.

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 Saffron 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 . . . 3
Summary . . . 5
Structure and Organization of Document . . . 6
Complete Analysis . . . 7
Code Coverage and Test Results for all files . . . 16
Tests written by Saffron team . . . 16
Tests written by Zokyo Secured team . . . 19

AUDITING STRATEGY AND TECHNIQUES APPLIED

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

Repository:
https://github.com/saffron-finance/spice-kcc-autocompounder/commit/bd95dd5b4ef963a820261102477eda7b8252bced

Last commit:
e8f256c6d24fd05bb84b3a278c6581c8edda24d7

Contracts:

- Saffron_Mojito_AdapterV2;
- IMojitoChef;
- ISaffron_Mojito_AdapterV2;
- IUniswapRouterV2;
- IUniswapV2Factory;
- IUniswapV2Pair;
- SaffronConverter;
- SaffronInsuranceFund;
- SaffronPoolV2;
- SaffronPositionNFT;
- SaffronPositionToken.

Throughout the review process, care was taken to ensure that the token contract:

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

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of Saffron 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:

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

SUMMARY

The Zokyo team has conducted a security audit of the given codebase. The contracts provided for an audit are well written and structured. All the findings within the auditing process are presented in this document.

Worth mentioning that we didn't find any critical issue. During the auditing process, our auditor's team found 1 issue with a high severity level, 7 issues with a low severity level, and 6 informational issues. Not all issues were resolved by the Saffron team.

Based on the results of the audit, we can give a score of 97 and state that the audited contracts are fully production-ready.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of 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. 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 ability of the contract to compile or operate in a significant way.

● 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

No autocompound execution in function get_holdings() at Saffron_Mojito_Adapter

HIGH | RESOLVED

According to comments for function get_holdings() it should execute autocompound() from SaffronMojitoAutocompounder but currently, it only returns the adapter amount staked in Mojito Swap. In current implementation autocompound() is not executed from any of audited SC.

Recommendation:
We advise to add autocompound functionality to function get_holdings as it described in comments or if you have another method for execution of autocompound change function description.

Re-audit:
Fixed.

Duplication of require statements

LOW | UNRESOLVED

At the current state of Saffron_Mojito_AdapterV2.sol, SaffronConverter.sol, SaffronInsuranceFund.sol, SaffronPoolV2.sol some of the require statements are duplicated.

Recommendation:
Consider to use Access manager or Ownable from OpenZeppelin to avoid duplication of require statements all over the contracts or make modifiers for repeated checks.

Re-audit:
Saffron team considered this exhibit but opted to retain the existing behaviour in the codebase.

Variable Mutability Optimization at SaffronPositionNFT.sol and SaffronPositionToken.sol

LOW | UNRESOLVED

The pool variable is only set once within the codebase's constructor.

Recommendation:
We advise it to be set as immutable optimizing the codebase's gas cost.

Re-audit:
Saffron team considered this exhibit but opted to retain the existing behaviour in the codebase.

Two identical functions at SaffronMojitoAutocompounder

LOW | UNRESOLVED

Function get_autocompounder_holdings() and get_mojito_chef_holdings() returns user info for the auto compounder.

Additionally, function get_autocompounder_holdings() at SaffronMojitoAutocompounder can be restricted to view.

Recommendation:
We advise to leave just one function with this functionality.

Re-audit:
Saffron team considered this exhibit but opted to retain the existing behaviour in the codebase.

Variable Mutability Optimization at SaffronPoolV2

LOW | UNRESOLVED

The variables base_asset, position_token, and NFT are only set once within the codebase's constructor.

Recommendation:
We advise it to be set as immutable optimizing the codebase's gas cost.

Re-audit:
Saffron team considered this exhibit but opted to retain the existing behaviour in the codebase.

Lock pragma to a specific version

LOW | RESOLVED

At the current state of all contracts being audited pragma is set to version range ^0.8.0 and it is better to lock the pragma to a specific version since not all the EVM compiler versions support all the features, especially the latest one's which are kind of beta versions, So the intended behavior written in code might not be executed as expected.

Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs.

Recommendation:
Lock pragma to a specific version.

Re-audit:
Fixed.

Function with commented logic at SaffronPositionNFT.sol

LOW | RESOLVED

Function renew() has commented logic and never called from SaffronInsuranceFund.sol.

Recommendation:
Consider either to use this function from SaffronInsuranceFund.sol or remove it if not needed.

Re-audit:
Fixed.

Unused variable at SaffronPoolV2

LOW | RESOLVED

In function update_exchange_rate() you use for calculation fee manager earnings reminder of total earnings and senior tranche earnings. Variable F_frac is never used.

Recommendation:
Remove F_frac to optimize the codebase's gas cost.

Re-audit:
Fixed.

Order of Layout, order of functions and naming conventions

INFORMAIONAL | UNRESOLVED

The contract elements should be grouped and ordered in the following way:

- Pragma statements;
- Import statements;
- Interfaces;
- Libraries;
- Contract.

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

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

The functions are not grouped according to their visibility and order. 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.

Ordering and naming conventions helps readers to navigate the code and find the elements more quickly.

Recommendation:
Consider changing order and name conventions according to solidity documentation: Solidity docs.

Re-audit:
Saffron team considered this exhibit but opted to retain the existing behaviour in the codebase.

Debugger import and Hardhat's console.log is on audit commits

INFORMATIONAL | RESOLVED

At the current state of all contracts being audited you are using hardhat debugging tools. It will increase code readability if removing unnecessary debugging tools from final code.

Recommendation:
Remove unnecessary debugging tools from audited commit.

Re-audit:
Fixed.

Return URI for unexciting tokenId

INFORMATIONAL | RESOLVED

Function tokenURI() will always return the same URI even for unexciting tokenId's.

Recommendation:
Consider adding a check for the existence of tokenId.

Re-audit:
Fixed.

Not indexed address parameters in events

INFORMATIONAL | RESOLVED

Consider a possibility to make addresses in events indexed. From [solidity docs](#):

You can add the attribute indexed to up to three parameters which adds them to a special data structure known as “topics” instead of the data part of the log. A topic can only hold a single word (32 bytes) so if you use a reference type for an indexed argument, the Keccak-256 hash of the value is stored as a topic instead.

All parameters without the indexed attribute are ABI-encoded into the data part of the log.

Recommendation:
Consider to adding indexed to events which you plan to filter with help of FE or BE.

Re-audit:
Fixed.

Unused local variable at SaffronConverter.sol

INFORMATIONAL | RESOLVED

In function remove_liquidity_would_return_zero() you declared unused local variable blocktime. If this return value not needed you can change your code as:

```
(uint256 balance0, uint256 balance1,) = IUniswapV2Pair(_pair_address).getReserves();
```

Recommendation:
Remove unused variable.

Re-audit:
Fixed.

Misleading comment at line 22 of SaffronPoolV2

INFORMATIONAL | RESOLVED

According to the comment, the version should be 2 but the constant variable version is equal to 0.

Recommendation:
Change version number.

Re-audit:
Fixed.

	Saffron_Mojito_AdapterV2; IMojitoChef; ISaffron_Mojito_AdapterV2; IUniswapRouterV2; IUniswapV2Factory; IUniswapV2Pair;	SaffronConverter; SaffronInsuranceFund; SaffronPoolV2; SaffronPositionNFT; SaffronPositionToken.
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

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Saffron team

Code Coverage

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

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts\	98.51	81.41	96.08	98.52	
SaffronConverter.sol	97.75	76.09	100.00	97.83	172, 185
SaffronInsuranceFund.sol	100.00	86.67	100.00	100.00	
SaffronPoolV2.sol	98.99	86.67	100.00	98.96	193
SaffronPositionToken.sol	95.00	62.50	75.00	95.00	71
contracts\adapters\	98.41	94.74	100.00	100.00	
Saffron_Mojito_AdapterV2.sol	98.41	94.74	100.00	100.00	
All files	**98.52**	**84.02**	**97.62**	**98.81**	

Test Results

```
Lorem
  ✓ should deploy adapter contract and autocompounder contract correctly (5572ms)
  ✓ should deploy SaffronPoolV2 contract correctly (5201ms)
  ✓ SaffronPoolV2 should deploy SaffronPositionToken contract correctly
  ✓ should have Saffron_Mojito_Adapter set_pool work correctly (2507ms)
  ✓ should have Saffron_Mojito_Adapter deploy_capital work correctly (13777ms)
  ✓ should have Saffron_Mojito_Adapter return_capital work correctly (1921ms)
  ✓ should have adapter set_lp work correctly (114ms)
  ✓ should have autocompounder set_mojito_chef work correctly (110ms)
  ✓ should have adapter propose_governance work correctly (62ms)
```

```
    ✓ should have adapter accept_governance work correctly (66ms)
    ✓ should have autocompounder propose_governance work correctly (58ms)
    ✓ should have autocompounder accept_governance work correctly (82ms)
    ✓ should have autocompounder set_autocompound_enabled work correctly (103ms)
    ✓ should have adapter sweep_erc work correctly (80ms)
    ✓ should have autocompounder sweep_erc work correctly (95ms)
    ✓ should have autocompounder emergency_withdraw work correctly (91ms)
    FATAL ERROR: latest_pool_holdings < yield_receiver_supply  0 800000000000000000000
    ✓ should have SaffronPoolV2 deposit work correctly (3092ms)
    ✓ should have autocompounder SFI swap all of tokensRewards for tokenA (1708ms)
    ✓ should have SaffronPoolV2 withdraw work correctly (82ms)
    ✓ should have SaffronPoolV2 set_exchange_rate work correctly (232ms)
    ✓ should have SaffronPoolV2 withdraw_fees work correctly (232ms)
    ✓ should have SaffronPoolV2 shut_down_pool work correctly (89ms)
    ✓ should have SaffronPoolV2 disable_deposits work correctly (82ms)
    ✓ should have SaffronPoolV2 propose_governance work correctly (57ms)
    ✓ should have SaffronPoolV2 accept_governance work correctly (62ms)
    ✓ should have SaffronPoolV2 set_adapter work correctly (76ms)
    ✓ should have SaffronPositionToken mint work correctly
    ✓ should have SaffronPositionToken burn work correctly
    ✓ should give us info on the state of the contract (47757ms)
    ✓ should have SaffronPoolV2 sweep_erc work correctly (99ms)
    ✓ should have SaffronMojitoAutocompounder reset_approvals work correctly (272ms)
    ✓ should have SaffronMojitoAutocompounder blend work correctly (51ms)
    ✓ should have SaffronMojitoAutocompounder spill work correctly

  test everything again
    ✓ tries to cause an error by setting pool to 0 lp then depositing (1305ms)

  test the insurance fund contract
    ✓ deploys the insurance fund contract and hooks it up to the pool (7306ms)
    ✓ deposits to the insurance fund contract (12307ms)
    ✓ earns some interest on the pool and checks the state of the pool then begins to
      unfreeze NFTs (44391ms)
    ✓ tests emergency_withdraw (59ms)
    ✓ withdraws nfts intermittently (1683ms)
    ✓ fund emergency withdraw and pending_earnings should work correctly (2442ms)
    ✓ should have fund propose_governance work correctly (72ms)
    ✓ should have fund accept_governance work correctly (64ms)
    ✓ should have fund set_treasury work correctly (57ms)
    ✓ should have fund set_pool work correctly (57ms)
    ✓ should test the conversions function by forcing uniswap to swap small amounts (8747ms)
    ✓ should remove all assets from pool / autocompounder (1809ms)

  multiple consecutive deposit/withdraw tests
    ✓ should have random users depositing and withdrawing randomly return correct
      amounts (123557ms)
    ✓ should have random users depositing and withdrawing randomly return correct
      amounts (156439ms)
    ✓ should have random users depositing and withdrawing randomly return correct
      amounts (94994ms)
    ✓ should have random users depositing and withdrawing randomly return correct
      amounts (105556ms)

  50 passing (11m)
```

Tests written by Zokyo Security team

As part of our work assisting Saffron 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 Saffron contract requirements for details about issuance amounts and how the system handles these.

Code Coverage

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

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts\	97.01	89.10	100.00	97.03	
SaffronConverter.sol	96.63	78.26	100.00	96.74	152, 165, 170
SaffronInsuranceFund.sol	92.86	86.67	100.00	92.98	122, 124, 127, 128
SaffronPoolV2.sol	98.98	95.00	100.00	98.95	168
SaffronPositionNFT.sol	100.00	100.00	100.00	100.00	
SaffronPositionToken	100.00	100.00	100.00	100.00	71
contracts\adapters\	98.41	97.37	100.00	100.00	
Saffron_Mojito_AdapterV2.sol	98.41	97.37	100.00	100.00	
contracts\interfaces\	100.00	100.00	100.00	100.00	
IMojitoChef.sol	100.00	100.00	100.00	100.00	
ISaffron_Mojito_AdapterV2.sol	100.00	100.00	100.00	100.00	
IUniswapRouterV2.sol	100.00	100.00	100.00	100.00	
IUniswapV2Factory.sol	100.00	100.00	100.00	100.00	
IUniswapV2Pair.sol	100.00	100.00	100.00	100.00	
All files	**97.28**	**90.72**	**100.00**	**97.58**	

Test Results

```
Contract: SaffronConverter
  Functions:
    init_conversions
      ✓ should revert if caller is not governance (917ms)
      ✓ should revert if the lengths of the arrays are different or equal to zero (341ms)
      ✓ should revert if the percentage is more than 100 (720ms)
      ✓ should revert if the operation is invalid (334ms)
      ✓ should init conversions correctly (14852ms)
    propose_governance
      ✓ should revert if caller is not governance (261ms)
      ✓ should revert if new governance is zero address (218ms)
      ✓ should propose governance correctly (797ms)
    accept_governance
      ✓ should revert if caller is not new governance (273ms)
      ✓ should accept governance correctly (389ms)
    sweep_erc
      ✓ should revert if caller is not governance (286ms)
      ✓ should transfer swept assets correctly (3152ms)

Contract: SaffronPoolV2
  constructor
    ✓ cannot deploy with zero adapter address (3017ms)
    ✓ cannot deploy with zero base asset token address (2810ms)
  Functions:
    deposit
      ✓ should revert if deposits disabled (495ms)
      ✓ should revert if the amount is zero (462ms)
      ✓ should revert if pool shut down (270ms)
      ✓ should revert if insufficient balance (439ms)
      ✓ should deposit correctly (18464ms)
      ✓ should catch event
    withdraw
      ✓ should revert if pool shut down (456ms)
      ✓ should revert if insufficient balance (530ms)
      ✓ should withdraw correctly (1442ms)
      ✓ should catch event
```

```
shut_down_pool
  ✓ should revert if caller is not governance (276ms)
  ✓ should shut down pool correctly (694ms)
  ✓ should catch event
disable_deposits
  ✓ should revert if caller is not governance (242ms)
  ✓ should disable deposits correctly (365ms)
  ✓ should catch event
begin_unfreeze
  ✓ should revert if pool shut down (461ms)
  ✓ should revert if not insurance token (5770ms)
  ✓ should begin unfreeze correctly (4645ms)
  ✓ should catch event
propose_governance
  ✓ should revert if caller is not governance (290ms)
  ✓ should revert if new governance is zero address (278ms)
  ✓ should set new governance correctly (375ms)
  ✓ should catch event
accept_governance
  ✓ should revert if caller is not new governance (225ms)
  ✓ should set governance correctly (318ms)
  ✓ should catch event
set_fee_manager
  ✓ should revert if caller is not governance (239ms)
  ✓ should revert if fee_manager is a zero address (290ms)
  ✓ should set fee manager correctly (857ms)
set_adapter
  ✓ should revert if caller is not governance (239ms)
  ✓ should revert if new adapter is a zero address (262ms)
  ✓ should set adapter correctly (688ms)
  ✓ should catch event
sweep_erc
  ✓ should revert if caller is not governance (228ms)
  ✓ should sweep funds correctly (1789ms)
  ✓ should catch event
set_exchange_rate
  ✓ should revert if pool not shut down (250ms)
  ✓ should revert if caller is not governance (513ms)
```

```
        ✓ should set exchange rate correctly (1027ms)
        ✓ should catch event
      withdraw_fees
        ✓ should revert if caller is not governance or fee manager (310ms)
        ✓ should revert if withdraw to zero address (302ms)
        ✓ should withdraw fees correctly (2081ms)
        ✓ should catch event

  Contract: SaffronInsuranceFund
    constructor
      ✓ cannot deploy with zero insurance asset address (7521ms)
      ✓ cannot deploy with zero base asset token address (5716ms)
    Functions:
      deposit
        ✓ should revert if pool shut down (2029ms)
        ✓ should revert if deposits disabled (634ms)
        ✓ should revert if the deposit amount is zero (300ms)
        ✓ should deposit correctly (12440ms)
        ✓ should catch event
      withdraw
        ✓ should revert if pool shut down (742ms)
        ✓ should revert if withdraw non-insurance NFT (725ms)
        ✓ should withdraw correctly (4597ms)
        ✓ should catch event
      emergency_withdraw
        ✓ should revert if pool shut down (913ms)
        ✓ should revert if withdraw non-insurance NFT (292ms)
        ✓ should revert if caller is not NFT owner (2206ms)
        ✓ should emergency withdraw correctly (2046ms)
        ✓ should catch event
      total_principal
        ✓ should return total amount of Insurance Assets correctly (280ms)
      set_pool
        ✓ should revert if caller is not governance (296ms)
        ✓ should set pool correctly (30267ms)
      set_treasury
        ✓ should revert if caller is not governance (389ms)
        ✓ should revert if new treasury is zero address (233ms)
        ✓ should set new treasury address correctly (636ms)
      pending_earnings
        ✓ should return pending earnings for a given Saffron Position NFT correctly (4636ms)

  Contract: SaffronMojitoAutocompounder
    constructor
      ✓ cannot deploy with zero adapter address (488ms)
      ✓ cannot deploy with zero LP address (1083ms)
      ✓ cannot deploy with zero router address (2478ms)
      ✓ cannot deploy with zero Mojito Staking contract address (3760ms)
      ✓ should deploy with correct adapter address (837ms)
      ✓ should deploy with correct LP address (120ms)
      ✓ should deploy with correct Mojito Staking contract address (111ms)
      ✓ should approve to Mojito Staking contract correctly (191ms)
    Functions:
      reset_approvals
        ✓ should reset LP token approvals correctly (1153ms)
      blend
        ✓ should revert if caller is not adapter (238ms)
        ✓ should send LP assets to Mojito Staking contract correctly (6174ms)
      spill
        ✓ should revert if caller is not adapter (364ms)
        ✓ should send LP assets from Mojito Staking to recipient correctly (10432ms)
        ✓ should withdraw from MojitoChef and return funds to router correctly (1470ms)
      set_mojito_chef
        ✓ should revert if caller is not governance (307ms)
        ✓ should set new MojitoChef contract address correctly (708ms)
      set_autocompound_enabled
        ✓ should revert if caller is not governance (415ms)
        ✓ should enable and disable auto compounding correctly (787ms)
      emergency_withdraw
        ✓ should revert if caller is not governance (351ms)
        ✓ should transfer LP assets from Mojito Staking correctly (15248ms)

  Contract: SaffronPositionToken
    constructor
      ✓ should deploy with correct token name (103ms)
      ✓ should deploy with correct token symbol (101ms)
      ✓ should deploy with correct Saffron pool address (126ms)
    Functions:
      mint
        ✓ should mint tokens correctly (2377ms)
        ✓ should revert if caller is not a pool (264ms)
      burn
        ✓ should burn tokens correctly (357ms)
        ✓ should revert if caller is not a pool (523ms)
      transfer
        ✓ should transfer tokens correctly (987ms)
        ✓ should revert if sender is a zero address (214ms)
        ✓ should revert if recipient is a zero address (245ms)
        ✓ should revert if transfer amount exceeds sender's balance (306ms)
      transferFrom
        ✓ should transfer tokens correctly (1159ms)
        ✓ should revert if sender is a zero address (225ms)
        ✓ should revert if recipient is a zero address (399ms)
        ✓ should revert if transfer amount exceeds sender's balance (269ms)
      approve
        ✓ should approve tokens correctly (643ms)
        ✓ should revert if approve to the zero address (251ms)
      allowance
        ✓ should return allowance correctly (85ms)
        ✓ should increase allowance correctly (284ms)
        ✓ should decrease allowance correctly (375ms)
        ✓ should revert if subtracted amount more than current allowance (228ms)

  Contract: SaffronPositionNFT
    constructor
      ✓ should deploy with correct token name (176ms)
      ✓ should deploy with correct token symbol (172ms)
      ✓ should deploy with correct Saffron pool address (103ms)
    Functions:
      mint
        ✓ should mint token correctly (7037ms)
        ✓ should revert if caller is not a Saffron pool or fund (293ms)
      burn
        ✓ should revert if unfreezing time has not expired (810ms)
        ✓ should burn token correctly (1878ms)
        ✓ should revert if caller is not a Saffron pool or fund (234ms)
      begin_unfreeze
        ✓ should begin unfreeze correctly (4782ms)
        ✓ should revert if caller is not a Saffron pool (231ms)
        ✓ should revert if user is not a owner of token (358ms)
        ✓ should revert if token already unfreezing (242ms)
      set_insurance_fund
        ✓ should begin unfreeze correctly (296ms)
        ✓ should revert if caller is not a Saffron pool (380ms)
        ✓ should revert if new insurance fund is a zero address (228ms)
      baseURI
        ✓ should return base URI correctly (76ms)
      tokenURI
        ✓ should return token URI correctly (92ms)

  Contract: saffron_Mojito_Adapter
    constructor
      ✓ cannot deploy with zero lp address (486ms)
    Functions:
      deploy_capital
        ✓ should revert if caller is not pool (406ms)
        ✓ should add funds to underlying protocol correctly (2804ms)
        ✓ should catch event
      return_capital
        ✓ should revert if caller is not pool (766ms)
        ✓ should return funds to user correctly (1168ms)
        ✓ should catch event
      get_holdings / get_holdings_view
        ✓ should return balance of Auto Compounder holdings correctly (2280ms)
        ✓ should catch event
      set_autocompounder
        ✓ should revert if caller is not governance (275ms)
        ✓ should set autocompander address correctly (1160ms)
      set_pool
        ✓ should revert if caller is not governance (408ms)
        ✓ should revert if pool is a zero address (260ms)
        ✓ should set pool address correctly (835ms)
      set_lp
        ✓ should revert if caller is not governance (179ms)
        ✓ should set mlp address correctly (816ms)
      set_lp
        ✓ should revert if caller is not governance (259ms)
        ✓ should set mlp address correctly (631ms)
      propose_governance
        ✓ should revert if caller is not governance (372ms)
        ✓ should revert if new governance is zero address (196ms)
        ✓ should set new governance correctly (283ms)
      accept_governance
        ✓ should revert if caller is not new governance (400ms)
        ✓ should set governance correctly (1203ms)
      sweep_erc
        ✓ should revert if caller is not governance (289ms)
        ✓ should sweep funds correctly (1454ms)
        ✓ should catch event

  165 passing (7m)
```

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

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.

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