StrategyImperamaxLender Audit: Team 3 (Minamoto)

---

1. Initialize function is external without modifier

The `initialize` function is external and has no modifier limiting who can call this function. The `initialize` function can only be called once. The `initialize` function call could be frontrun right after the StrategyImperamaxLender.sol contract is deployed to initialize the contract with unintended values. This would result in

Proof of concept/Steps to Reproduce

After the StrategyImperamaxLender.sol contract is deployed, frontrun the call to `initialize` with custom values.

Impact

This attack would require redeployment of StrategyImperamaxLender.sol, which would result in minor gas loss and some annoyance. If the attack is persistent and happens multiple times, it could delay the contract deployment.

Risk Breakdown

Low. The deploy.py main function should return an error because the contract can only be initialized once.

Recommendation

Initializing in the constructor would be ideal, but may not be feasible due to the need to clone the contract.

—

2. `_initializeStrat` called twice but duplicate pools not checked

The `_initializeStrat` function is called from the `constructor` and the `initialize` functions. If the `constructor` adds some pools to the "pools" state variable and the `initialize` function adds the same pools, there is no check for duplicate pools. Contrast this to the `addTarotPool` function, which loops through all the new pools to confirm there are no duplicates being added to the `pools` state variable.

Proof of concept/Steps to Reproduce

If the same "_pools" array of non-zero length is provided to the `constructor` and `initialize` functions, each pool would exist in the "pools" array twice https://github.com/dudesahn/StrategyImperamaxLender/blob/379061675e1549fb2b8ebcf23218abd3729ae06c/contracts/StrategyImperamaxLender.sol#L67

This is prevented in the `addTarotPool` function with a require statement https://github.com/dudesahn/StrategyImperamaxLender/blob/379061675e1549fb2b8ebcf23218abd3729ae06c/contracts/StrategyImperamaxLender.sol#L499-L502

Impact

Medium

Risk Breakdown

Duplicate pools in the "pools" array could cause logic errors, but the functions that modify this variable should only be called by authorized users

Recommendation

Add the same for loop from `addTarotPool` to `initialize` or `_initializeStrat` would prevent duplicate pools from getting added. An easier option is to call `_initializeStrat` from the constructor using an empty array instead of the "_pools" array

—

3. trueExchangeRate doesn't handle zero case

The `trueExchangeRate` function does not have a special code branch for the case where either "actualBalance" or "totalSupply" is zero. While this edge case may be rare, there is a special if statement to handle this case in the Tarot PoolToken.sol `exchangeRate` function and the Tarot Borrowable.sol `exchangeRate` function.

Proof of concept/Steps to Reproduce

The `trueExchangeRate` function does not handle the edge case of actualBalance == 0 or totalSupply == 0:
https://github.com/dudesahn/StrategyImperamaxLender/blob/379061675e1549fb2b8ebcf23218abd3729ae06c/contracts/StrategyImperamaxLender.sol#L135

But the Tarot PoolToken.sol code handles it:
https://github.com/tarot-finance/tarot-core/blob/4cd572e62c07411320ce690d254767bd14a414a2/contracts/PoolToken.sol#L47

And the Tarot Borrowable.sol code handles it:
https://github.com/tarot-finance/tarot-core/blob/4cd572e62c07411320ce690d254767bd14a414a2/contracts/Borrowable.sol#L83

Impact

I don't see a way for this to be malicious unless there is some way for a user to withdraw all the funds of a low liquidity pool for a temporary denial of service of some sort.

Risk Breakdown

Low

Recommendation

Handle the zero case with special logic like Tarot does if it could cause unforeseen edge cases

—

4. safeApprove when adding a pool

When the `addTarotPool` function is called, a pool is added to the "pools" state variable, but the pool is not approved by calling `want.safeApprove`. Contrast this to `_initializeStrat` which calls `want.safeApprove(_pools[i], type(uint256).max)`. Either the safeApprove is missing in one of these places or it should be removed from the other.

Proof of concept/Steps to Reproduce

`safeApprove` is not called when a new pool is added in `addTarotPool`
https://github.com/dudesahn/StrategyImperamaxLender/blob/379061675e1549fb2b8ebcf23218abd3729ae06c/contracts/StrategyImperamaxLender.sol#L495-L505

`safeApprove` is called when a new pool is added in `_initializeStrat`
https://github.com/dudesahn/StrategyImperamaxLender/blob/379061675e1549fb2b8ebcf23218abd3729ae06c/contracts/StrategyImperamaxLender.sol#L74

Impact

One of these two functions is wrong and I suspect `safeApprove` can be removed from `_initializeStrat` because I don't see where a pool would call `transferFrom` to withdraw from this strategy contract

Risk Breakdown

Medium, approving when not necessary could result in token loss if approved contract has a vulnerability

Recommendation

Remove safeApprove (or, if I am wrong, add safeApprove consistently)

—

5. Initialize pools count at 1 instead of 0, view calls on potentially infinitely growing array.

Steps to Reproduce

Manual: L87 & L100 in array of pools.

Impact

May cause issues with overflow / underflow with for-loops used in multiple functions impacting arrays, especially with pool reordering.

Recommendation

Functions calling view on an indefinitely growing array should be avoided; push pool by one to prevent any pool from getting ID 0.

6. withdraw(), liquidatePosition() may require reentrancyGuard

Steps to reproduce

Manual

Impact

Allows withdrawal without changing of liquidated amount

Recommendation

Implement reentrancyGuard to withdraw() and liquidatePosition() functions

7. prepareReturn() does not handle 0 case

Steps to Reproduce

Manual

Impact

When assets = debt, there may be an edge case. Where prepareReturn() can have a 0 and any functions calling prepareReturn may revert as a result.

Recommendation

Correctly handle 0 cases.

8. Require statements comes AFTER the transfer in deposit()

Steps to Reproduce

Manual

Impact

Gas savings: The earlier the function can revert as a result of the require being earlier in the function, the more gas we can save

Recommendation

Implement require before the .transfer() on L319. In addition, safeTransfer should be used.

9. attemptToRemovePool (and several other functions) include .transfer(), can use .safeTransfer()

Steps to Reproduce

Manual

Impact

Low - for more safety with token transfers.

Recommendation

Implement safeTransfer (and all other occurences of .transfer with safeTransfer()).