OPTIMUM
BLOCKCHAIN SECURITY

Yearn Finance

Smart contract
Security Assessment
Notional Finance Strategy

March, 2022

Table Of Contents

Disclaimer

Overview Page

Summary

Project name	Yearn Finance
URL	https://yearn.finance/
Code	https://github.com/jmonteer/notional-lending-strategy/tree/16slim/basic_strategy
Commit hash	8a5a9bf5a114aab36e835bb4d8b1d6ba75b8a234
Mitigations commit hash	e3edeca9eeb6518c83ce2f553284d1d14b4e445e
Language	Solidity

Contracts Assessed

Contract name	SHA-1
/contracts/Strategy.sol	8460bb541ed9ef81bda3b0e82ab0a731eee23267

Findings Summary

Severity	Found	Resolved	Partially resolved	Acknowledged
High	0	0	0	0
Medium	0	0	0	0
Low	4	3	0	1
Informational	0	0	0	0
Total	4	3	0	1

Classification of issues

Severity	
High	Vulnerabilities that may directly result in loss of funds, and thus require an urgent fix.
Medium	Issues that may not be directly exploitable, or with a limited impact, are still required to be fixed.
Low	Subjective issues with a negligible impact.
Informational	Subjective issues or observations with negligible or no impact.

Findings

Issue #01	**Violation of the "checks-effects-interactions" pattern**
Severity	Low
Location	Strategy.sol 1. _checkPositionsAndWithdraw (L971) 2. adjustPosition (L522) 3. liquidatePosition (L755)
Description	Writing to storage variables after (untrusted) external calls may potentially lead to reentrancy attack vectors. Although the impact of such attacks might be limited in this case, it is still recommended to follow the "checks-effects-interactions" pattern, where writing operations ("effects") precede external calls ("interactions").
Recommendation	*1. maturity = 0* can be moved a line before the call to *nProxy.withdraw* (L966). *2. maturity = minMarketMaturity* can be moved to L475. *3. toggleRealizeLosses = false* can be moved to L642.
Resolution	Fixed in e3edeca9eeb6518c83ce2f553284d1d14b4e445e by adopting the auditor's recommendations.

Issue #02	Unsafe castings
Severity	Low
Location	Strategy.sol: 1. *_getTotalValueFromPortfolio* (L998, L992, L1002) 2. *adjustPosition* (L490) 3. *liquidatePosition* (L688, L700) 4. *_initializeNotionalStrategy* (L146) 5. *getTradeFrom* (L535, L536) 6. *_rollOverTrade* (L1118)
Description	There are many unsafe casting operations in the code, it is hard to assess the exact impact of these. In general, it may lead to wrong code execution which will lead to unintended results.
Recommendation	Consider using SafeCast, or alternatively, check that the value to be cast is inside the bounds of the type. In case gas efficiency is important, consider just proving that a harmful casting operation is not possible, and add a code comment for that.
Resolution	Fixed in e3edeca9eeb6518c83ce2f553284d1d14b4e445e by using SafeCast, and also by rejecting "out of bounds" values for int88.

Issue #03	Possible negation of the most negative number
Severity	Low
Location	Strategy.sol: 1. *adjustPosition* (L498) 2. *liquidatePosition* (L688, L698) 3. *_getTotalValueFromPortfolio* (L998)
Description	For the compiler version used in this repo, negating the most negative number will result in the same number, instead of its positive equivalent.
Recommendation	Consider bumping the compiler version to >=0.8.0, or alternatively, revert for MIN_INT cases.
Resolution	Acknowledged, MIN_INT cases are highly unlikely, since *amountToTrade* is based on 8 decimals, and the rest are int256 numbers.

Issue #04	**Potential integer underflow**
Severity	Low
Location	Strategy.sol: *_getMinimumMarketIndex* (L1053)
Description	*_getMinimumMarketIndex* subtracts the current *block.timestamp* from *_activeMarkets[i].maturity* unsafely. *nProxy.getActiveMarkets* should return results where *_activeMarkets[i].maturity >= block.timestamp*. While it seems to be the case in the current version of *nProxy*, it should not be counted upon solely, since Notional contracts are upgradeable.
Recommendation	Consider bumping the compiler version to >=0.8.0, or alternatively, use *SafeMath.sub*.
Resolution	Fixed in e3edeca9eeb6518c83ce2f553284d1d14b4e445e by using *SafeMath.sub.*

Trust Assumptions

	Notional finance **smart contracts**
Description	*Notional finance* contracts are trusted in many ways, including but not only: ● contracts in use are upgradeable. ● Funds deposited into the protocol are effectively locked until maturity, exiting a position before maturity will add the cost of borrowing.

Observations

1. *nProxy* value is injected in the initialization phase, which violates Yearn’s production policy, specifying that addresses should be hardcoded instead.
2. Loops are used in multiple places in the code. More specifically, *_accountPortfolio* and *_activeMarkets* are being iterated over. That may lead to a denial of service caused by the block gas limit. Currently, the intention is to have a maximum of 3 positions in the account portfolio and 3 active markets. It is highly recommended to include tests to measure the number of iterations that will cause these transactions to revert.
3. Current code does not support multiple maturities. Although there are many places in the code where loops are used to iterate, in practice only one variable is used to store the maturity of the current position.

Recommendations

Recommendation #01	Gas optimizations
Description	*Strategy._getTotalValueFromPortfolio* - line 989 get executed in each iteration of the inner loop, which can be moved outside of the inner loop, to save some gas.

Recommendation #02	*Strategy._rollOverTrade* - Outdated natspec
Description	The natspec documentation for this function describes a return value, where in practice no value is returned. Consider updating the natspec comments for this function.

Recommendation #03	*Strategy.liquidateAllPositions* - Handling the case of *_marketIndex == 0*
Description	*_getMarketIndexForMaturity* might return 0 for *_marketIndex* which will end up reverting in *CashGroup.loadMarket*. However, this case is handled differently in *liquidatePosition* and *_rollOverTrade*. Consider adopting a consistent policy to handle this edge case.

OPTIMUM

BLOCKCHAIN SECURITY