OPTIMUM
BLOCKCHAIN SECURITY

Yearn Finance

Smart contract
Security Assessment
Stargate Strategy

May, 2022

Table Of Contents

Disclaimer

This report does not provide any security warranty, investment advice, endorsement, or disapproval of any particular project or team. This report does not provide a warranty that the code in scope is completely free of vulnerabilities, bugs, or potential exploits. This report does not assess the financial risk of any asset. No third party should rely on this report to make any decisions to buy or sell any asset or product.

Delivering secured code is a continuous challenge that requires multiple steps. It is strongly recommended to use best code practices, write a full test suite, conduct an internal audit, and launch a bug bounty program as a complement to this report.

It is the sole responsibility of the project team to ensure that the code in scope is functioning as intended and that the recommendations presented in this report are carefully tested before deployment.

Overview Page

Summary

Project name	Yearn Finance
URL	https://yearn.finance/
Code	https://github.com/OpenOrg-gg/yearn-stargate
Commit hash	b113405d720a7997aefcb34ee2a5caea8ea3a1a6
Mitigations commit hash	
Language	Solidity

Contracts Assessed

Contract name	SHA-1
/contracts/Strategy.sol	acdcca85aafc3cca8c9799a75d9432b3a98c41c7

Findings Summary

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

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	***Strategy._withdrawFromLP* - *instantRedeemLocal* may not withdraw the entire *_lpAmount* specified**
Severity	Medium
Location	*Strategy.sol* - *_withdrawFromLP* (L402)
Description	Stargate is using a complex system of cross-chain liquidity management, where liquidity is scattered among different chains. Using *stargateRouter.instantRedeemLocal* may not be enough to withdraw the entire LP position.
Recommendation	Consider improving the logic of *_withdrawFromLP* by calling multiple different functions of *stargateRouter* to support a better withdrawal operation.
Resolution	

Issue #02	**ERC20 approvals of type(uint256).max**
Severity	Low
Location	*Strategy.sol* - *_initializeThis* (L107)
Description	Approving the maximum value of uint256 is a known practice to save gas. However, this pattern was proven to increase the impact of an attack many times in the past, in case the approved contract gets hacked.
Recommendation	Consider approving the exact amount that's needed to be transferred, or alternatively, add an external function that allows the revocation of approvals.

Resolution	

Issue #03	***Strategy.claimAndSellRewards, prepareReturn*** **- Potential “sandwiching” front-running vectors**
Severity	Low
Location	*Strategy.sol* 1. *claimAndSellRewards* *2. prepareReturn*
Description	*_sellRewardsUniv3* is using *amountOutMinimum = 0*, and *_sellRewardsCurve* is using an on-chain price quote, both are susceptible to “sandwiching” attacks where the front-runner can order the transactions so that he will be able to buy the asset on a different AMM #1, sell it on the current AMM that is used by the strategy (denoted by AMM #2), to push the price down, then include the transaction that calls either *claimAndSellRewards* or *prepareReturn* which will push the price even lower, then the front-runner can buy the asset on AMM #2 and sell it back on AMM #1.
Recommendation	While both *claimAndSellRewards* and *prepareReturn* are susceptible, *prepareReturn* is harder to mitigate, since adding another parameter is impossible due to the *Strategy* interface. *prepareReturn* is called in the context of *harvest* which is called by a keeper bot using flashbots API, that is slashed if preforming a sandwich attack, which reduces this risk signaficantly. The risk is left open for *claimAndSellRewards*, however, and thus it is recommended to add a *minAmountOut* parameter to *claimAndSellRewards.*
Resolution	

Observations

1. Addresses used as external dependencies are injected in the initialization phase, which violates Yearn's production policy, specifying that addresses should be hardcoded instead.

Recommendations

Recommendation #01	**Gas optimizations**
Description	*Strategy.withdrawSome* - *balanceOfStakedLPToken* is called twice (lines 276-277) instead of once.

OPTIMUM

BLOCKCHAIN SECURITY