OPTIMUM
BLOCKCHAIN SECURITY

Yearn Finance

Smart contract
Security Assessment

Tokemak Strategy

February, 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/red-impala/tokemak-eth-strat/blob/master/contracts/Strategy.sol
Commit hash	d22005318d25cc2cea41318131c35c1392620534
Mitigations commit hash	a5675a46f356abd0e139c85dfa7ab29b981b9ea0
Language	Solidity

Contracts Assessed

Contract name	SHA-1
/contracts/Strategy.sol	ef2fbab0b16be9e668f4d9ab334d8be0617e5aed

Findings Summary

Severity	Found	Resolved	Partially resolved	Acknowledged
High	0	0	0	0
Medium	0	0	0	0
Low	2	1	0	1
Informational	0	0	0	0
Total	2	1	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, that 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._checkAllowance* approves *tokemakEthPool* to spend *type(uint256).max***
Severity	Low
Location	Strategy.sol L289
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 instead.
Resolution	Fixed in a5675a46 by approving the exact amount.

Issue #02	***Strategy.requestWithdrawal* - The total time of a withdrawal might be lengthen**
Severity	Low
Location	Strategy.sol L221
Description	According to *Tokemak* documentation and implementation, a call to *requestWithdrawal* will always overwrite the current request, and thus might lengthen the total waiting time.
Recommendation	Consider adding a check that there are no active withdrawal requests before calling *tokemakEthPool.requestWithdrawal*.
Resolution	Acknowledged but not fixed, since the scenario of extending the total waiting time is less probable.

Trust Assumptions

Tokemak smart contracts	
Description	The *Tokemak* contracts are trusted in many ways, including but not only: • The *Tokemak* contracts in use are upgradeable. • Funds deposited in *Tokemak* pools are withdrawable only after 24 hours. The user should Ensure to complete any outstanding withdrawal before he requests another. Not doing so will overwrite the current request and he will have to wait again for those funds. • Tokemak's *Rewards* contract admin can set the *rewardsSigner* which will effectively cancel any unclaimed rewards signed by the previous signer.

Observations

1. In current implementation, the *want* token of *Strategy* can't be generalized to tokens that are not backed 1:1 by ETH.
2. *tokemakEthPool* might be paused by the owner, which will cause *BaseStrategy.harvest / tend* to revert.
3. Withdrawing the funds deposited to *tokemakEthPool* is possible only after a call to *tokemakEthPool.requestWithdrawal*, which is currently restricted for *onlyEmergencyAuthorized*, instead of being part of the *Strategy.liquidatePosition* flow.

Recommendations

Recommendation #01	**Gas optimizations**
Description	*Strategy.prepareReturn* - return values from *liquidatePosition* are not used, and instead, these values are generated again in *prepareReturn*.

Recommendation #02	***Strategy.prepareMigration - SafeERC20 usage***
Description	*SafeERC20* is not used in this function, although practically it is not an issue, consider using *safeTransfer* to maintain code best practices uniformity.

Recommendation #03	**Change the package name**


OPTIMUM
BLOCKCHAIN SECURITY