Audit of the YFValue Protocol

a report authored by

Minh Khai Do

Summaries by

Rasikh Morani

Joel Farris

innovative fortuna iuvat

29 August, 2020

0.1 Executive Summary

A Representative Party of the YFValue Decentralized Organization ("YFValue") engaged The Arcadia Group, to conduct a review of the YFValue Smart Contracts ("YFV Protocol"). Arcadia performed this engagement for a period of one week from August 24th, 2020 to August 28th, 2020. With a total of one engineer and one overseeing project manager, the audit was successfully completed in the estimated timeframe. The review was focused on common security flaws alongside potentially introduced vulnerabilities.

Arcadia completed the audit using various methods primarily consisting of dynamic and static analysis. The assessment identified a small number of issues, ranging in areas of code quality and health, although no high or critical severity issues were found.

0.2 Recommendations Summary

0.2.1 Short Term

- Remediate all known findings
- Implement scaling deposit and daily volume limits to de-risk smart contracts, alternatively, break larger pools into multiple contracts to spread holdings across multiple contracts
- Move all present and future contracts to having the governance multisig as the *Owner*

Findings

Dynamic Findings

Severity:

Medium

Contracts:

$YFV_Rewards.sol, YFV_Rewards_Pool1_BAL.sol, YFV_Rewards_Pool2_YFI.sol,$
$YFV_Rewards_Pool3_BAT.sol, YFV_Rewards_Pool4_REN.sol, YFV_Rewards_Pool5_KNC.sol,$
$YFV_Rewards_Pool6_BTC.sol, YFV_Rewards_Pool7_ETH.sol, YFV_Rewards_Pool8_LINK.sol,$
$YFV_Rewards_Pool9_YCrvUNIv2.sol$

Type:

Dynamic

Lines:

Line 735-737

Description:

The rewardStake function can claim all of the YFV tokens from the pool, when owner makes a new rewardStake, the new address can withdraw all of the YFV tokens within the pool

Severity:

Optimization

Contract:

YFV$_V ote.sol$

Type:

Dynamic

Lines:

Line 253-260

Category:

Logic issue

Description:

Inefficient function, when the contract is identifying the minimum staking power, it loops through the verification process too often, leading to heavily increased gas costs.

Severity:

Low–Code Optimization

Contract:

YFV$_S$*take.sol*

Type:

Dynamic

Lines:

Line 680-688 (675-683, 807-830)

Description:

Modifier is used to set rewards data, while this practice is somewhat common in smart contracts, changing it to an internal function would lead to a lowered risk

Static Findings

Static Analysis can yield useful information that may be irrelevant in the context of the contracts.

Severity:

Low-Code Optimization

Contract:

All Contracts

Type:

Static

Description:

YFV utilizes a floating pragma, which is not recommended per SWC-103, which recommends utilizing a fixed pragma to avoid potential introduced issues, and so the bytecode does not vary between builds.

Severity:

Low

Contract:

All Pool Contracts

Type:

Static

Lines:

847-849

Description:

YFV utilizes multiple writes to a persistent state following an external call, which may lead to the potential introduction of re-entrancy vulnerabilities. It is recommended to implement re-entrancy locks if possible.

Severity:

Low

Contract:

All Pool Contracts

Type:

Static

Lines:

847

Description:

YFV utilizes multiple calls within a single transaction, while this may not be an issue in this specific situation, it is recommended to exercise caution and per SWC-113 follow a code style that completes only one external call per transaction, or ensure that all callees are trusted.

Severity:

Notice

Contract:

All Pool Contracts

Type:

Static

Lines:

808 (on /yfv/yfv$_{r}ewards_{p}ool1_{b}al.solcontract$)

Description:

YFV utilizes a the block.timestamp function and while it is useful in identifying the next epoch, it is important to note that this introduces a certain reliance on miners.