19th FEBRUARY 2021

SMART CONTRACT AUDIT REPORT

-

version v2.0

Smart Contract Security Audit and General Analysis

HAECHI AUDIT

Table of Contents

1 Issues (0 Critical, 0 Major, 1 Minor) Found

About HAECHI AUDIT

HAECHI AUDIT is a global leading smart contract security audit and development firm operated by HAECHI LABS. HAECHI AUDIT consists of professionals with years of experience in blockchain R&D and provides the most reliable smart contract security audit and development services.

So far, based on the HAECHI AUDIT's security audit report, our clients have been successfully listed on the global cryptocurrency exchanges such as Huobi, Upbit, OKEX, and others.

Our notable portfolios include SK Telecom, Ground X by Kakao, and Carry Protocol while HAECHI AUDIT has conducted security audits for the world's top projects and enterprises.

Trusted by the industry leaders, we have been incubated by Samsung Electronics and awarded the Ethereum Foundation Grants and Ethereum Community Fund.

Contact : audit@haechi.io
Website : audit.haechi.io

01. Introduction

This report was written to provide a security audit for the yAxis MetaVault v2 smart contract. HAECHI AUDIT conducted the audit focusing on whether yAxis MetaVault v2 smart contract is designed and implemented in accordance with publicly released information and whether it has any security vulnerabilities.

The issues found are classified as CRITICAL, MAJOR, MINOR or TIPS according to their severity.

CRITICAL	Critical issues are security vulnerabilities that MUST be addressed in order to prevent widespread and massive damage.
MAJOR	Major issues contain security vulnerabilities or have faulty implementation issues and need to be fixed.
MINOR	Minor issues are some potential risks that require some degree of modification.
TIPS	Tips could help improve the code's usability and efficiency

HAECHI AUDIT advises addressing all the issues found in this report.

02. Summary

The code used for the audit can be found at GitHub

- https://github.com/yaxis-project/metavault
  - Commit Hash : 4c5431ba4f631c5b10dff37468ea8b4743cb62fd

Update

- [v2.0] Commit hash 3ad508e772c51ae04369d7aeab70b58794069d6d included fixes for 1 Minor issue.

Issues

HAECHI AUDIT has 0 Critical Issues, 0 Major Issues, and 1 Minor Issue; also, we included 0 Tip category that would improve the usability and/or efficiency of the code.

Severity	Issue	Status
MINOR	StableSwap3PoolConverter, yAxisMetaVaultManager#governanceRecoverUnsupported will fail to transfer for some tokens	(Found - v1.0) (Resolved - v2.0)
Notice	Governance address can control user's fund	(Found - v1.0)

03. Overview

Contracts Subject to Audit

- StrategyControllerV2.sol
- StableSwap3PoolConverter.sol
- BaseStrategy.sol
- StrategyCurve3Crv.sol
- StrategyDforce.sol
- StrategyPickle3Crv.sol
- yAxisMetaVault.sol
- yAxisMetaVaultHarvester.sol
- yAxisMetaVaultManager.sol

Roles

The yAxis MetaVault v2 Smart contract has the following authorizations:

- **governance**
- **strategist**

The features accessible by each level of authorization is as follows:

Role	Functions
governance	● StrategyControllerV2 ○ addStrategy ○ claimInsurance ○ setVaultManager ○ inCaseStrategyGetStuck ○ inCaseTokensGetStuck ○ removeStrategy ○ reorderStrategies ○ setCap ○ setConverter ○ setInvestEnabled ○ setMaxStrategies ○ setVault ○ withdrawAll ○ harvestStrategy ● StableSwap3PoolConverter ○ governanceRecoverUnsupported ○ convert

    - convert_stables
- BaseStrategy
    - approveForSpender
    - setController
    - setRouter
    - deposit
    - harvest
    - skim
    - withdraw
    - withdrawAll
- yAxisMetaVault
    - setMin
    - setGovernance
    - setController
    - setCoverter
    - setVaultManager
    - setEarnLowerlimit
    - setTotalDepositCap
    - setAcceptContractDepositor
    - setYaxPerBlock
    - setEpochEndBlock
    - setEpochRewardMultipler
    - setTreasuryWallet
    - claimInsurance
    - earnExtra
- yAxisMetaVaultManager
    - governanceRecoverUnsupported
    - setGovernance
    - setInsuranceFee
    - setInsurancePool
    - setInsurancePoolFee
    - setStakingPool
    - setStakingPoolShareFee
    - setStrategist
    - setTreasury
    - setTreasuryBalance
    - setTreasuryFee
    - setWithdrawalProtectionFee
    - setYax
    - setControllerStatus

	○ setHarvester ○ setVaultStatus ● yAxisMetaVaultHarvester ○ addStrategy ○ removeStrategy ○ setController ○ setHarvester ○ setVaultManager
strategist	● StrategyControllerV2 ○ inCaseTokensGetStuck ○ removeStrategy ○ reorderStrategies ○ setCap ○ setConverter ○ setInvestEnabled ○ setMaxStrategies ○ setVault ○ withdrawAll ○ harvestStrategy ● BaseStrategy ○ deposit ○ harvest ○ skim ○ withdraw ○ withdrawAll ● yAxisMetaVaultManager ○ setControllerStatus ○ setHarvester ○ setVaultStatus ● yAxisMetaVaultHarvester ○ addStrategy ○ removeStrategy ○ setController ○ setHarvester ○ setVaultManager

Notice

- **Governance address can control user’s fund**

  As stated above in section “Overview”, governance address and strategist address has access to change many values. This can result in withdrawing the user's asset.

  Make sure the governance address is a contract address and keep on eye what is being processed through governance.

04. Issues Found

MINOR : StableSwap3PoolConverter, yAxisMetaVaultManager#governanceRecoverUnsupported will fail to transfer for some tokens. (Found - v1.0) (Resolved - v2.0)

MINOR

Problem Statement

In StableSwap3PoolConverter, yAxisMetaVaultManager#governanceRecoverUnsupported() will fail to transfer tokens that do not return bool type value(ex. usdt)

Recommendation

Use SafeERC20 library for token transfer.

Update

[v2.0] - yAxis team has applied recommendation to their code by using SafeERC20 library and also fully tested with off-standard tokens.

05. Disclaimer

This report is not an advice on investment, nor does it guarantee adequacy of a business model and/or a bug-free code. This report should be used only to discuss known technical problems. The code may include problems on Ethereum that are not included in this report. It will be necessary to resolve addressed issues and conduct thorough tests to ensure the safety of the smart contract.

Appendix A. Test Results

The results below show the unit test results that cover the main logic of the smart contract subject to the security audit. The parts marked in red are test cases that failed to pass the test due to issues.

```
BaseStrategy
 ✓ should not allow unpermissioned callers (112ms)
 ✓ should approve tokens for spending
 ✓ should skim stuck tokens out of the strategy
 ✓ should send the insurancePoolFee to the insurancePool (423ms)
 ✓ should set the controller
 ✓ should set the router
 COVERED BY HAECHI
  ✓ cannot be deployed with invalid values
  withdraw
   ✓ do not change amount if balance is larger than amount
   ✓ should revert if vault address is zero
  withdrawAll
   ✓ should revert if vault address is zero
  _payHarvestFees
   ✓ should not accure stking fee if staking pool is not set
   ✓ should not accure treasury fee if treasury is not set
   ✓ should not accure insurance fee if insurance is not set

StableSwap3PoolConverter
 ✓ should not allow unpermissioned callers (168ms)
 ✓ should approve for spender (84ms)
 ✓ should set the vault manager (150ms)
 ✓ should set the StableSwap3Pool (223ms)
 COVERED BY HAECHI
  ✓ should revert for non authorized callers
  convert
   ✓ should not revert when output != token3CRV && input != token3CRV (60ms)
  convert_rate
   ✓ should return zero if output != token3CRV && input != token3CRV
  calc_token_amount
   ✓  should return appropriate amount
  calc_token_amount_withdraw
   ✓  should return appropriate amount
   ✓ should return zero for non-pool token
```

```
    governanceRecoverUnsupported
      ✓ should withdraw token

  StrategyControllerV2
    ✓ should deploy with expected state
    ✓ should not allow unpermissioned callers (177ms)
    ✓ should add a strategy (65ms)
    ✓ should deposit into first strategy (507ms)
    ✓ should obey maximum strategies amount (44ms)
    ✓ should add an additional strategy (57ms)
    ✓ should deposit into second strategy (302ms)
    ✓ should reorder strategies (58ms)
    ✓ should withdraw excess funds when reducing a strategy cap (96ms)
    ✓ should deposit into first strategy when cap of second is reached (352ms)
    ✓ should withdraw small amounts (460ms)
    ✓ should deposit large amounts into a single strategy (305ms)
    ✓ should withdraw large amounts from multiple strategies (306ms)
    ✓ should remove strategies (97ms)
    ✓ should deposit/earn to the remaining strategy (253ms)
    ✓ should allow all strategies to be removed (67ms)
    ✓ should allow deposits without strategies (212ms)
    ✓ should earn to a newly added strategy (125ms)
    ✓ should harvest strategy through controller (220ms)
   COVERED BY HAECHI
    addStrategy
      ✓  should revert if strategy is already active
      ✓  should revert if converter is zero address
    claimInsurance
      ✓ should revert if msg.sender is not governance
      ✓ should claim insurance
    setVaultManager
      ✓ should revert if msg.sender is not governance
      ✓ should change vault manager
    removeStrategy
      ✓ should revert if startegy is not active
      ✓ if _want != vault.want(), set strategy in coverter
    reorderStrategies
      ✓ should revert if strategy is same
      ✓ should revert if any strategy is inactive
    setCap
      ✓ should fail if strategy is not active
      ✓ should not withdraw if cap == 0
    ✓ should change cap
  setInvestEnabled
    ✓ should change investEnabled
  setMaxStrategies
    ✓ should revert if new max is zero
  setVault
    ✓ should revert if vault address of token is not zero
  investEnabled
    ✓ should return false when global invest enabled is false
  want
    ✓ should return appropriate want token
  withdrawFee
    ✓ should return appropriate value

StrategyCurve3Crv
  ✓ should deploy with initial state set
  ✓ should deposit DAI (258ms)
  ✓ should harvest (224ms)
  ✓ should withdraw to DAI (235ms)
  ✓ should withdrawAll to 3CRV (118ms)
  ✓ should deposit USDT (269ms)
  ✓ should withdrawAll by controller (43ms)
  COVERED BY HAECHI
   ✓ cannot be deployed with invalid values
   deposit
    ✓ do not interact with gauce when _wantBal is zero
   getMostPremium
    ✓ should choose dai if dai balance is lowest
    ✓ should choose usdc if usdc balance is lowest
    ✓ should choose usdt if usdt balance is lowest
   _harvest
    ✓ should do nothing is remaining weth is zero
    ✓ should not deposit if balanceOfWant is zero

StrategyDforce
  ✓ should deploy with initial state set
  ✓ should deposit DAI (436ms)
  ✓ should harvest (195ms)
  ✓ should withdraw to DAI (434ms)
  ✓ should withdrawAll to 3CRV (314ms)
  ✓ should deposit USDT (499ms)
  ✓ should withdrawAll by controller (185ms)
  COVERED BY HAECHI
    ✓ cannot be deployed with invalid values
    deposit
    ✓ do not interact with dToken when amount is zero
    ✓ do not stake if dtoken balance is zero
    _harvest
    ✓ should do nothing is remaining weth is zero
    ✓ should not deposit if balanceOfWant is zero
    _withdraw
    ✓ should not convert if amount is zero
    _withdrawAll
    ✓ should not redeem when dToken balance is zero
    _convert
    ✓ should revert if covert_rate is zero

StrategyFlamIncome
  ✓ should deploy with initial state set
  ✓ should deposit USDT (448ms)
  ✓ should withdraw to DAI (450ms)
  ✓ should withdrawAll to 3CRV (116ms)
  ✓ should deposit USDT (474ms)
  ✓ should withdrawAll by controller (202ms)

StrategyGenericVault
  ✓ should deploy with initial state set (39ms)
  ✓ should deposit USDT (465ms)
  ✓ should withdraw to DAI (476ms)
  ✓ should withdrawAll to 3CRV (115ms)
  ✓ should deposit USDT (479ms)
  ✓ should withdrawAll by controller (179ms)
  COVERED BY HAECHI
    ✓ cannot be deployed with invalid values
    deposit
    ✓ do not deposit if balanceOfWant is zero
    _harvest
    ✓ should do nothing
    _withdraw
    ✓ should not convert if amount is zero
    _withdrawAll
    ✓ should not redeem when vault balance is zero
    _convert
    ✓ should revert if covert_rate is zero
```

```
StrategyIdle
  ✓ should deploy with initial state set
  ✓ should deposit DAI (451ms)
  ✓ should harvest (209ms)
  ✓ should withdraw to DAI (489ms)
  ✓ should withdrawAll to 3CRV (337ms)
  ✓ should deposit USDT (449ms)
  ✓ should withdrawAll by controller (230ms)
  COVERED BY HAECHI
    ✓ cannot be deployed with invalid values
    deposit
      ✓ do not interact with idletoken when balance is zero
    _harvest
      ✓ should do nothing if remaining weth is zero
    _withdraw
      ✓ should not convert if amount is zero
    _withdrawAll
      ✓ should not convert if amount is zero
    _convert
      ✓ should revert if covert_rate is zero
    _liquidateAsset
      ✓ should do nothing if balance is zero

StrategyPickle3Crv
  ✓ should deploy with initial state set
  ✓ should deposit DAI (307ms)
  - should harvest
  ✓ should withdraw to DAI (272ms)
  ✓ should withdrawAll to 3CRV (151ms)
  ✓ should deposit USDT (302ms)
  ✓ should withdrawAll by controller (56ms)
  COVERED BY HAECHI
    ✓ cannot be deployed with invalid values
    setStableForLiquidity
      ✓ should revert if msg.sender is not authorized
      ✓ should revert if token is not registered
      ✓ should change values
    setPickleMasterChef
      ✓ should revert if msg.sender is not authorized
      ✓ should change values
    setPoolId
```

```
      ✓ should revert if msg.sender is not authorized
      ✓ should change values
    _deposit
      ✓ should not interact with pickle when balances are zero
    _addLiquidity
      ✓ should supply liquidity
    _harvest
      ✓ should not add liquidity if remaining weth is zero
    _withdraw
      ✓ should adjust amount if amount is larger than staked amount

StrategyStabilize
  ✓ should deploy with initial state set
  ✓ should deposit DAI (478ms)
  ✓ should harvest (205ms)
  ✓ should withdraw to DAI (430ms)
  ✓ should withdrawAll to 3CRV (310ms)
  ✓ should deposit USDT (476ms)
  ✓ should withdrawAll by controller (203ms)
  COVERED BY HAECHI
    ✓ cannot be deployed with invalid values
    calculateZPATokenWithdrawFee
      ✓ should reduce fee when feeSubstraction is larger than initial_fee + end_fee
    deposit
      ✓ do not interact with zpatoken when balance is zero
      ✓ do not interact with zpapool when balance is zero
    _harvest
      ✓ should do nothing if remaining weth is zero
      ✓ should do nothing if balanceOfWant is zero
    _withdraw
      ✓ should not convert if amount is zero
    _withdrawAll
      ✓ should not convert if amount is zero
    _convert
      ✓ should revert if covert_rate is zero

StrategyYearnV2
  ✓ should deploy with initial state set
  ✓ should deposit DAI (430ms)
  ✓ should withdraw to DAI (413ms)
  ✓ should withdrawAll to 3CRV (290ms)
  ✓ should deposit USDT (455ms)
  ✓ should withdrawAll by controller (177ms)
  COVERED BY HAECHI
    ✓ cannot be deployed with invalid values
    _harvest
      ✓ should do nothing
    _withdraw
      ✓ should not convert if amount is zero
    _withdrawAll
      ✓ should not convert if amount is zero
    _convert
      ✓ should revert if covert_rate is zero

StrategydYdXSoloMargin
  ✓ should deploy with initial state set
  ✓ should deposit DAI (448ms)
  ✓ should withdraw to DAI (423ms)
  ✓ should withdrawAll to 3CRV (507ms)
  ✓ should deposit USDT (463ms)
  ✓ should withdrawAll by controller (193ms)
  COVERED BY HAECHI
    ✓ cannot be deployed with invalid values
    _deposit
      ✓ should not interact with dydx if amount is zero
    _harvest
      ✓ should do nothing
    _withdraw
      ✓ should not convert if amount is zero
    _withdrawAll
      ✓ should not convert if amount is zero
    _convert
      ✓ should revert if covert_rate is zero
    balanceOfPool()
      ✓ should return 0 if balance.sign is zero

oracle_safety
  ✓ should be safe from the Yearn yDAI vault attack (424ms)

stuck_funds.test
  ✓ deposit (249ms)
  ✓ stuck WETH in strategy (54ms)
  ✓ stuck WETH in controller (40ms)
  ✓ stuck t3crv.address (core) in strategy (101ms)
```

```
yAxisMetaVault
  ✓ should deposit (334ms)
  ✓ should depositAll (217ms)
  ✓ should stakeShares (211ms)
  ✓ should pendingYax (41ms)
  ✓ should unstake(0) for getting reward (48ms)
  ✓ should unstake
  ✓ should withdraw T3CRV (39ms)
  ✓ should withdraw DAI (166ms)
  ✓ should withdraw USDT (167ms)
  ✓ should withdraw need unstake (205ms)
  ✓ should withdrawAll to USDC (373ms)
  COVERED BY HAECHI
    checkContract
      ✓ should accept contract when acceptContractDepositor is true
      ✓ should not accept contract whend acceptContractDepositor is false and msg.sender is contract
    balance
      ✓ should be able to handle when controller is zero address
    setGovernance
      ✓ should fail if msg.sender is not governance
      ✓ should change governance address
    setController
      ✓ should fail if msg.sender is not governance
    setConverter
      ✓ should fail if msg.sender is not governance
      ✓ should fail if converter cannot handler 3crv token
    setVaultManager
      ✓ should fail if msg.sender is not governance
    setEarnLowerLimit
      ✓ should fail if msg.sender is not governance
    setTotalDepositCap
      ✓ should fail if msg.sender is not governance
    setAcceptContractDepositor
      ✓ should fail if msg.sender is not governance
      ✓ should change acceptContractDepositor
    setYaxPerBlock
      ✓ should fail if msg.sender is not governance
      ✓ should update reward
      ✓ should update yaxperblock
    setEpochEndBlock
      ✓ should fail if msg.sender is not governance
      ✓ should fail if index is out of range
      ✓ should fail if epochendnumber is earlier than now
      ✓ should fail if epochEndBlock of index has expired
      ✓ should update epochEndBlock
    setEpochRewardMultiplier
      ✓ should fail if msg.sender is not governance
      ✓ should fail if index is out of range
      ✓ should fail if epochEndBlock of last index has expired
      ✓ should update epochRewardMultiplier
    getMultiplier
      ✓ should be able to get appropriate multiplier
    setTreasuryWallet
      ✓ should fail if msg.sender is not governance
      ✓ should change treasury wallet
    claimInsurance
      ✓ should release insurance if msg.sender is controller
      ✓ should fail if msg.sender!=controller && msg.sender !=governance
      ✓ should transfer token to treasury if msg.sender is governance
    withdrawFee
      ✓ should send fee to when controller is not zero address
    earn
      ✓ should not do anything when controller is zero address
    calc_token_amount_deposit
      ✓ should return appropriate value
    calc_token_amount_withdraw
      ✓ should return appropriate value
    convert_rate
      ✓ should return appropriate value
    deposit
      ✓ should revert if zero amount
      ✓ should do nothing when user deposited non relevant tokens
      ✓ should revert when deposit has exceeded depositCap
      ✓ should not revert when vaultmanager address is not zero address
      ✓ should accure insurance when insurancefee is set
      ✓ can handle when shrae == 0
    depositAll
      ✓ should deposit without convert on 3crv
      ✓ should not revert even convert rate is zero
      ✓ should be able to handle only 3crv deposit
      ✓ should revert when deposit has exceeded depositCap
      ✓ should for large slippage
      ✓ should be able to handle min_mint_amount ==0, and totalDeposit==0
```

```
      ✓ should not revert when vaultmanager address is not zero address
      ✓ should accure insurance when insurancefee is set
      ✓ can handle when shrae == 0
      ✓ should be able to deposit
    pendingYax
      ✓ can handle whend block.number is larger than lastReward block of lpSupply ==0
    updateReward
      ✓ should do nothing when lastRewardBlock is future
      ✓ should just update lastRewardBlock if lpSupply == 0
    harvest
      ✓ should fail if msg.sender is not controller
      ✓ should fail if reserve === 3crv
      ✓ should transfer token to controller
    unstake
      ✓ should fail if user.amount == _amount
    withdraw
      ✓ should fail if _need <= userInfo[msg.sender].amount
      ✓ can handle without vaultManager address
      ✓ do not accure withdrawalprotectionfee if not set
      ✓ can handle without controller address
      ✓ should fail if convert_rate == 0
    earnExtra
      ✓ should fail if msg.sender is not governance
      ✓ cannot transfer this or 3crv
      ✓ should fail if convert_rate is zero
      ✓ should convert to 3crv

  yAxisMetaVaultHarvester
    ✓ should not allow unpermissioned callers (78ms)
    ✓ should set the controller
    ✓ should set the vault manager
    ✓ should set harvesters
    ✓ should add strategies
    ✓ should harvest added strategies (179ms)
    ✓ should add additional strategies (50ms)
    ✓ should rotate harvesting strategies (202ms)
    ✓ should not allow harvestNextStrategy until timeout has passed (93ms)
    ✓ should remove strategies
    COVERED BY HAECHI
      removeStrategy
        ✓ do nothing when strategy is not found
```

```
yAxisMetaVaultManager
  ✓ should deploy with expected state (40ms)
  ✓ should not allow unpermissioned callers (97ms)
  ✓ should set the insurance fee
  ✓ should set the insurance pool
  ✓ should set the insurance pool fee
  ✓ should set the staking pool
  ✓ should set the staking pool fee
  ✓ should set the treasury
  ✓ should set the treasury balance
  ✓ should set the treasury fee
  ✓ should set the withdrawal protection fee
  ✓ should set the yax.address token
  ✓ should set the controller status
  ✓ should set the vault status
  ✓ should set the harvester
  ✓ should set the strategist
  ✓ should set the governance
  COVERED BY HAECHI
    governanceRecoverUnsupported
      ✓ should withdraw token
    setInsuranceFee
      ✓ should fail if above limit
    setInsurancePoolFee
      ✓ should fail if above limit
    setStakingPoolShareFee
      ✓ should fail if above limit
    setTreasuryFee
      ✓ should fail if above limit
    setWithdrawalProtectionFee
      ✓ should fail if above limit
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/					
StableSwap3PoolConverter.sol	100	100	100	100	
StableSwap3PoolOracle.sol	85	58.33	100	100	82,88,93
yAxisMetaVault.sol	100	100	100	100	

yAxisMetaVaultHarvester.sol	100	100	100	100	
yAxisMetaVaultManager.sol	100	100	100	100	
metavault/controllers/	100	100	100	100	
StrategyControllerV1.sol	100	100	100	100	
StrategyControllerV2.sol	100	100	100	100	
metavault/strategies/	100	100	100	100	
BaseStrategy.sol	100	100	100	100	
StrategyCurve3Crv.sol	100	100	100	100	
StrategyDforce.sol	100	100	100	100	
StrategyFlamIncome.sol	100	100	100	100	
StrategyGenericVault.sol	100	100	100	100	
StrategyIdle.sol	100	100	100	100	
StrategyPickle3Crv.sol	100	100	100	100	
StrategyStabilize.sol	100	100	100	100	
StrategyYearnV2.sol	100	100	100	100	
StrategydYdXSoloMargin.sol	100	100	100	100	