29th SEPTEMBER 2020

SMART CONTRACT AUDIT REPORT

version v1.0

Smart Contract Security Audit and General Analysis

HAECHI AUDIT

Table of Contents

6 Issues (0 Critical, 1 Major, 5 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 HarvestFinance smart contract. HAECHI AUDIT conducted the audit focusing on whether HarvestFinance 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/harvest-finance/harvest/). The last commit for the code audited is at "4f2812dc0765d402dc5e9685a015bd8b73b3d92b".

update

Harvest Finance team has answered that they have already fixed several issues on commit

- 48adf02d98b5bad2b426d7b833548aeddd62d2f7
- 8d464a1791a3d48d4b0318fb3c9207075cdede86
- 974a83ec3d5674f0c3f9b67ce015a1573462af3d

And they have said that they already acknowledged other issues.

Issues

HAECHI AUDIT has 2 Critical Issues, 1 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
MAJOR	CRVStrategyStable#depositArbCheck() always returns true	(Found - v1.0) (Fixed - 48ad)
MINOR	RewardPool#notifyRewardAmount() does not check if it received reward.	(Found - v1.0) (Acknowledged)
MINOR	RewardPool#notifyRewardAmount() can decrease rewardRate	(Found - v1.0) (Acknowledged)
MINOR	Vault#setVaultFractionToInvest() can not be set to enable full investment.	(Found - v1.0) (Fixed - 8d46)
MINOR	HardRewards#load() can lead to temporary loss of fund when changing token address	(Found - v1.0) (Acknowledged)
MINOR	NoMintRewardPool has an Owner which can be misleading against the Governor.	(Found - v1.0) (Acknowledged)

Notice	Governance role is not Contract and can move user’s funds without permission	(Found - v1.0) (Fixed - 974a)
Notice	Cannot add more minters	(Found - v1.0) (Acknowledged)

03. Overview

Contracts Subject to Audit

Roles

The HarvestFinance Smart contract has the following authorizations:

- **Governance**
- **Owner**
- **RewardDistribution**
- **HardWorker**

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

Role	Functions
Governance	• Storage ◦ setGovernance() ◦ setController() • HardRewards ◦ addVault() ◦ removeVault() ◦ load() • FeeRewardForwarder ◦ setTokenPool() • SNXRewardStrategy ◦ emergencyExit() ◦ continueInvesting() ◦ switchRewardSource() ◦ setRewardSource() ◦ setLiquidationRoute() ◦ salvage() • CRVStrategyYCRV ◦ salvage() • CRVStrategyWRenBTC ◦ setArbTolerance() ◦ salvage() ◦ setSell() ◦ setSellFloor() • CRVStrategyStable ◦ setArbTolerance()

	○ salvage() ○ setConvertor() ● CRVStrategySwerve ○ CRVStrategySwerve ○ setArbTolerance() ○ salvage() ○ setSell() ○ setSellFloor() ○ createLock() ○ checkPoint() ○ increaseAmount() ○ increaseUnlockTime() ○ withdrawLock() ● Controller ○ addHardWorker() ○ removeHardWorker() ○ addToGreyList() ○ removeFromGreyList() ○ setFeeRewardForwarder() ○ addVaultAndStrategy() ○ doHardWork() ○ rebalance() ○ setHardRewards() ○ salvage() ○ salvageStrategy() ● NotifyHelper ○ notifyPools() ● DelayMinter ○ announceMint() ○ excuteMint() ○ cancelMint() ○ setTeam() ○ setOperator() ○ renounceMinting() ● Governable ○ Governable ○ setStorage() ● Vault ○ doHardWork() ○ setStrategy() ○ rebalance() ○ withdrawAll() ○ setVaultFractionToInvest() ● RewardToken ○ addMinter() ● SNXRewardStrategy

	◦ withdrawAllToVault() ◦ withdrawToVault() ◦ doHardWork() • CRVStrategyYCRV ◦ withdrawAllToVault() ◦ withdrawToVault() ◦ doHardWork() • CRVStrategyWRenBTC ◦ withdrawAllToVault() ◦ withdrawToVault() ◦ doHardWork() • CRVStrategyStable ◦ withdrawAllToVault() ◦ withdrawToVault() ◦ doHardWork() • CRVStrategySwerve ◦ withdrawAllToVault() ◦ withdrawToVault() ◦ doHardWork()
Owner	• RewardPool ◦ renounceOwnership() ◦ transferOwnership() ◦ setRewardDistribution()
RewardDistribution	• RewardPool ◦ notifyRewardAmount()
HardWorker	• Controller ◦ doHardWork() ◦ rebalance()

Notice

- ### **Governance role is not Contract and can transfer user's funds without permission (Fixed - 974a)**

  As given on the above table, **Governance** can change addresses such as Storage and Strategy. Since Governance is not given as a contract, we assume that it is controlled as EOA that Harvest Finance team controls.

  This can lead to malfunction of the total system and even lead to transferring the user's fund without permission.

  Simple scenario would be,

  1. Deploy contract that pass Vault#setStrategy()'s requirements but does not act as Strategy like transferring the whole balance when doHardWork() is called.
  2. Call setStrategy() on Vault and change to
  3. call doHardWork() that actually sends whole balance to Governance

  This can be improved by,

  1. adding time lock to all functions that change address.
  2. Implementing Governance contract to restrict the action that Governance can do

  #### Update - 974a

  Harvest Finance has been working on timelocked upgrades and implemented on commit 974a. Since our audit was based on commit 4f28, it should be noted that the timelock upgrade is not audited.

- ### **RewardToken will not have any other minter (Acknowledged)**

  RewardToken#addMinter() checks if msg.sender is both **Minter** and **Governance**

  But we found that Governance added Minter to the DelayMinter contract they deployed, and renounced itself as Minter to ensure governance cannot add more minter.

Although it seems to be intended, it can lead to potential malfunction such as it cannot change the DealyMinter contract when needed.

Update

Harvest Finance has answered that this is intentional behavior.

04. Issues Found

MAJOR : CRVStrategyStable#depositArbCheck() always returns true

(Found - v.1.0) (Fixed - 48ad)

MAJOR

```
function depositArbCheck() public view returns(bool) {
  uint256 currentPrice = underlyingValueFromYCrv(ycrvUnit);
  if (currentPrice > curvePriceCheckpoint) {
    return currentPrice.mul(100).div(curvePriceCheckpoint) > 100 - arbTolerance;
  } else {
    return curvePriceCheckpoint.mul(100).div(currentPrice) > 100 - arbTolerance;
  }
}
```

Problem Statement

CRVStrategyStable#depositArbCheck() always returns true as the logic to choose calculation is written in the opposite way.

Recommendation

Change the inequality sign.

Update - 48ad

depositArbCheck() function has been updated to return a proper status

MINOR : RewardPool#notifyRewardAmount() does not check if it received reward. (Found - v.1.0) (Acknowledged)

MINOR

```
function notifyRewardAmount(uint256 reward)
    external
    onlyRewardDistribution
    updateReward(address(0))
{
    if (block.timestamp >= periodFinish) {
        rewardRate = reward.div(duration);
    } else {
        uint256 remaining = periodFinish.sub(block.timestamp);
        uint256 leftover = remaining.mul(rewardRate);
        rewardRate = reward.add(leftover).div(duration);
    }
    lastUpdateTime = block.timestamp;
    periodFinish = block.timestamp.add(duration);
    emit RewardAdded(reward);
}

```

Problem Statement

RewardPool#notifyRewardAmount() does not check if it has received the reward to distribute. It can lead to a high reward rate for farmers who get rewards faster than others. And can make others unable to earn the rewards.

Since this function is designed to be only called by rewardDistribution, this error can only be done by rewardDistribution.

Recommendation

Receive reward token by transferFrom when function is called.

Update

Harvest Finance team has answered that they are aware of this issue and they calculate reward diligently. Since the issue will not occur if the function is used with care and Harvest Finance team does not want to migrate contract to fix this issue, no further action needed for this issue. But it should be noted that the issue exists.

MINOR : RewardPool#notifyRewardAmount() can decrease rewardRate (Found - v.1.0) (Acknowledged)

MINOR

```
function notifyRewardAmount(uint256 reward)
    external
    onlyRewardDistribution
    updateReward(address(0))
{
    if (block.timestamp >= periodFinish) {
        rewardRate = reward.div(duration);
    } else {
        uint256 remaining = periodFinish.sub(block.timestamp);
        uint256 leftover = remaining.mul(rewardRate);
        rewardRate = reward.add(leftover).div(duration);
    }
    lastUpdateTime = block.timestamp;
    periodFinish = block.timestamp.add(duration);
    emit RewardAdded(reward);
}
```

Problem Statement

RewardPool#notifyRewardAmount() does not check if the rewardRate decreases after notification. Since it updates rate to be *(leftoverRate + notified reward)/duration* when previous reward is not finished, if rewardDistribution keeps notifying with zero reward, it can lead to continuous decrease on reward rate,

Since this function is designed to be only called by rewardDistribution, this error can only be done by rewardDistribution.

Recommendation

Check if rewardRate increases after notifying reward.

Update

Harvest Finance team has answered that they are aware of this issue and they calculate reward diligently. Since the issue will not occur if the function is used with care and Harvest Finance team does not want to migrate contract to fix this issue, no further action needed for this issue. But it should be noted that the issue exists.

MINOR : Vault#setVaultFractionToInvest() can not be set to enable full investment. (Found - v.1.0) (Fixed - 8d46)

MINOR

```
144.  function setVaultFractionToInvest(uint256 numerator, uint256 denominator) external
      onlyGovernance {
145.    require(denominator > 0, "denominator must be greater than 0");
146.    require(numerator < denominator, "denominator must be greater than numerator");
147.    vaultFractionToInvestNumerator = numerator;
148.    vaultFractionToInvestDenominator = denominator;
149.  }
150.
```

Problem Statement

Because of the line 146, vault can not invest the full amount of deposits

Recommendation

Change require statement to include when numerator is same as denominator

Update - 48ad

setVaultFractionToInvest() function has been updated to be able to enable full investment

MINOR : HardRewards#load() can lead to temporary loss of fund when changing token address (Found - v.1.0) (Acknowledged)

MINOR

```
function load(address _token, uint256 _rate, uint256 _amount) external onlyGovernance {
  token = IERC20(_token);
  blockReward = _rate;
  if (address(token) != address(0) && _amount > 0) {
    token.safeTransferFrom(msg.sender, address(this), _amount);
  }
}
```

Problem Statement

HardRewards#load() changes the token address that will be used to reward the hard workers. But, when the token address is changed, it does not give back the original token which can be resolved by changing back to original token, but this could lead to malfunction if other hardworker is rewarded between changes.

Recommendation

Transfer original token back to controller or governance when token address has changed.

Update

Harvest Finance team has answered that they do not have any plan to change the reward token.

MINOR : NoMintRewardPool has an Owner which can be misleading against the Governor. (Found - v.1.0) (Acknowledged)

MINOR

Problem Statement

NoMintRewardPool inherits Controllable and Owable which makes it has both Owner and Governance.
Since other contracts can be controlled by Controllable which checks storage that acts as a registry to track the governance address.
By adding Owner on RewardPool can lead to complexity in operation

Recommendation

Do not inherit the Ownable on NoMintRewardPool

Update

Harvest Finance team has answered that their intention is to minimize the modification on SNX rewardPool smart contract. Since the only function that needs owner privilege is setRewardDistribution(), it could be handled easily.

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.