9th November 2020

SMART CONTRACT AUDIT REPORT

-

version v2.0

Smart Contract Security Audit and General Analysis

HAECHI AUDIT

Table of Contents

0 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 PickleFinance smart contract. HAECHI AUDIT conducted the audit focusing on whether PickleFinance 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/pickle-finance/protocol/). The last commit for the code audited is at "9b0f330a16bc35c964211feae3b335ab398c01b6".

Issues

HAECHI AUDIT has 0 Critical Issues, 0 Major Issues, and 1 Minor Issue; also, we included 3 Tip category that would improve the usability and/or efficiency of the code.

Severity	Issue	Status
MINOR	Numerators can be larger than denominators.	(Found - v1.0) (Resolved - v2.0)
TIPS	Controller#revokeStrategy() can be called on active strategy	(Found - v1.0) (Resolved - v2.0)
TIPS	Unused Variables	(Found - v1.0) (Resolved - v2.0)
TIPS	Unused import	(Found - v1.0) (Acknowledged - v2.0)
Notice	Governance role can transfer user's funds without permission	(Found - v1.0) (Resolved - v2.0)

03. Overview

Contracts Subject to Audit

- timelock.sol
- controller-v3.sol
- pickle-jar.sol
- pickle-swap.sol
- staking-rewards.sol
- uni-curve-converter.sol
- masterchef.sol
- pickle-token.sol
- PicklesInTheCitadel.sol
- strategy-cmpd-dai-v1.sol
- strategy-base.sol
- strategy-curve-base.sol
- strategy-staking-rewards-base.sol
- strategy-uni-farm-base.sol
- crv-locker.sol
- scrv-voter.sol
- strategy-curve-3crv-v1.sol
- strategy-curve-rencrv-v1.sol
- strategy-curve-scrv-v3_1.sol
- strategy-curve-scrv-v4.sol
- strategy-curve-scrv-v4_1.sol
- strategy-uni-eth-dai-lp-v3_1.sol
- strategy-uni-eth-usdc-lp-v3_1.sol
- strategy-uni-eth-usdt-lp-v3_1.sol
- strategy-uni-eth-wbtc-lp-v1.sol

Roles

The PickleFinance Smart contract has the following authorizations:

- **Governance**
- **Strategist**
- **Owner**

- **Voter**
- **Keeper**

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

Role	Functions

Governance	• PickleJar
	  ○ setMin()
	  ○ setGovernance()
	• ControllerV3
	  ○ setJar()
	  ○ setConverter()
	  ○ setDevFund()
	  ○ setTreasury()
	  ○ setStrategist()
	  ○ setSplit()
	  ○ setOneSplit()
	  ○ setGovernance()
	  ○ setStrategy()
	  ○ inCaseTokensGetStuck()
	  ○ inCaseStrategyTokenGetStuck()
	  ○ yearn()
	  ○ withdrawAll()
	• StrategyBase
	  ○ setStrategist()
	• StrategyCurveBase
	  ○ setKeepCRV()
	• StrategyCurveSCRVv4
	  ○ setDevFundFee()
	  ○ setTreasuryFee()
	  ○ setPerformanceFee()
	  ○ setStrategist()
	  ○ setGovernance()
	  ○ setController()
	  ○ setKeepCRV()
	• CRVLocker
	  ○ addVoter()
	  ○ removeVoter()
	  ○ createLock()
	  ○ increaseAmount()
	  ○ increaseUnlockTime()
	  ○ release()
	  ○ setGovernance()
	  ○ execute()
	• SCRVVoter

	○ setGovernance() ○ approveStrategy() ○ revokeStrategy() ● StrategyCurveSCRVv4_1 ○ setKeepCRV() ○ setStrategist() ○ setGovernance() ● StrategyCmpdDaiV1 ○ addKeeper() ○ removeKeeper() ○ setColFactorLeverageBuffer() ○ setColFactorSyncBuffer()
Strategist	● ControllerV3 ○ setJar() ○ setConverter() ○ setStrategy() ○ withdrawAll() ○ inCaseTokensGetStuck() ○ inCaseStrategyTokenGetStuck() ○ yearn() ● StrategyCmpdDaiV1 ○ addKeeper() ○ removeKeeper() ○ setColFactorLeverageBuffer() ○ setColFactorSyncBuffer()
Owner	● MasterChef ○ add() ○ set() ○ setPicklePerBlock() ○ setBonusEndBlock() ○ setDevFundDivRate() ● StakingRewards ○ notifyRewardAmount() ○ recoverERC20() ○ setRewardsDuration()
Voter	● CRVLocker ○ withdraw() ○ createLock() ○ increaseAmount() ○ increaseUnlockTime() ○ release()

	○ execute()
Keeper	● StrategyCmpdDaiV1 ○ deleverageUntil() ○ leverageUntil()

Notice

- **Governance role can transfer user's funds without permission**

  As stated above, the Governance role has privilege to execute many functions including setting other roles which even makes it possible to withdraw user's funds without permission. Although Governance role is expected to be a Gnosis Safe contract[1] In most contracts of the pickle.finance, it should be noted that Gnosis Safe will not guarantee the safety of the user's funds.

  Following scenario is the example of Governance withdrawing user's funds without timelock

  1. deploy a malicious Controller contract that returns Governance's address on treasury(), devfund(), jars(). And add one external function that calls StrategyCurveSCRVv4#withdraw().
  2. Governance changes controller of StrategyCurveSCRVv4 to deployed malicious Controller
  3. Controller calls StrategyCurveSCRVv4#withdraw()
  4. Since StrategyCurveSCRVv4#withdraw() distributes deposit to treasury(), devfund(), jars(), it will send all funds to Governance since it is set to Governance address in (1)

  This can be prevented by rewriting duplicate codes. Inheriting StrategyBase and not overriding setContoller() will prevent this in code level

[1] https://etherscan.io/address/0x9d074e37d408542fd38be78848e8814afb38db17#readProxyContract

Update

[v2.0] StrategyCurveSCRVv4 has been deprecated and replaced by StrategyCurveSCRVv4_1 and using StrategyBase#setController(). Which makes it impossible to perform the attack described above.

04. Issues Found

MINOR : Numerators can be larger than denominators. (Found - v1.0)(Resolved - v2.0) MINOR

Controller#setSplit() and PickleJar#setMin() bot sets numerator used to calculate percentage of token amount. Although this issue will only occur when Governance executes functions with bad inputs, it will lead to contracts to revert on normal interactions.

Recommendation

Modify function to revert if numerators are larger than denominators.

Update

[v2.0] Pickle finance team has added a check for given numerators.

TIPS: Controller#revokeStrategy() can be called on active strategy (Found - v1.0)(Resolved - v2.0) TIPS

Controller#revokeStrategy() is used to revoke strategy to prevent old or buggy strategy to be active. Since revoke should mean that the strategy is no longer used, it will be clear to check if the strategy is registered as strategies[_token].

Update

[v2.0] Pickle finance team has added requirements that makes active strategy cannot be revoked.

TIPS: Unused Variables (Found - v1.0)(Resolved - v2.0) TIPS

Controller#strategyConverters is defined but not used in contract. Consider removing the variables.

Update

[v2.0] Pickle finance team has moved to a new version of Controller(V4) and variable strategyConverters is removed.

TIPS: Unused import (Found - v1.0)(Acknowledged - v2.0) TIPS

strategy-curve-scrv-v3_1.sol imports scrv-voter.sol and crv-locker.sol but it can be compiled without these imports. Consider removing these imports.

Update

[v2.0] Pickle finance team is planning to update the code to use the unused import.

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.