QuillAudits

Audit Report
September, 2021

Contents

Scope of Audit

The scope of this audit was to analyze and document the veToken smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	0	1	0
Closed	0	0	3	6

Introduction

From **August 25, 2021 to August 30, 2021**- QuillAudits Team performed a security audit for veToken Finance smart contracts.

The code for the audit was taken from following the official links:
https://github.com/vetoken-finance/vetoken-contracts/blob/main/contracts/pickle/PickleBaseRewardPool.sol

https://github.com/vetoken-finance/vetoken-contracts/blob/main/contracts/pickle/PickleBooster.sol

https://github.com/vetoken-finance/vetoken-contracts/blob/main/contracts/pickle/PickleDepositor.sol

Branch: main
Initial Audit Commit hash: 3ed5fc8f60f16f05f5eb2139e8b02c78436ee5a6
Revised Audit Commit hash: fef3c2aa7cd2a6613d9c38ae7edff108518fa826

Issues Found – Code Review / Manual Testing

A. Contract - PickleBaseRewardPool.sol

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low level severity issues

1. **Missing events at state changing functions**

   Description
   It is recommended to emit an event whenever there is a state changing function. This is useful to the clients to track the actions appropriately. Following functions should emit an event.
   - addExtraReward
   - clearExtraRewards
   - updateReward
   - donate

   Remediation
   We suggest adding events in the above functions.

   Status: Fixed

2. **Infinite loop possibility**

Line	Code

159	for (uint256 i = 0; i < extraRewards.length; i++) {
	IRewards(extraRewards[i]).stake(msg.sender, _amount);
	}

182	for (uint256 i = 0; i < extraRewards.length; i++) {
	IRewards(extraRewards[i]).stake(_for, _amount);
	}

Line	Code

201	for (uint256 i = 0; i < extraRewards.length; i++) {
	IRewards(extraRewards[i]).withdraw(msg.sender, amount);
	}

228	for (uint256 i = 0; i < extraRewards.length; i++) {
	IRewards(extraRewards[i]).withdraw(msg.sender, amount);
	}

265	for (uint256 i = 0; i < extraRewards.length; i++) { IRewards(extraRewards[i]).getReward(_account);

Description
The array extraRewards is used in loops, without having any limits. If those addresses are too many, then this might hit the block's gas limit.

Remediation
We suggest putting some array length limit, or reward manager should add limited wallets only.

Status: Acknowledged by the Auditee

Informational

3. Consider using latest solidity version

pragma solidity 0.6.12;

Description
We recommend using the latest solidity version as there are so many breaking changes, and security optimization has been implemented in the latest solidity version.

Remediation
Consider using solidity version 0.8.7, which is the latest at the time of this audit.

Status: Fixed

B. Contract - PickleBooster.sol

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low level severity issues

1. **Missing events at state changing functions**

   Description
   It is recommended to emit an event whenever there is a state-changing function. This is useful to the clients to track the actions appropriately. Following functions should emit an event.
   - setOwner
   - setFeeManager
   - setPoolManager
   - setFactories
   - setArbitrator
   - setVoteDelegate
   - setRewardContracts
   - setFees
   - setTreasury
   - addPicklePool
   - shutdownPool
   - shutdownSystem
   - vote

   Remediation
   We suggest adding events in the above functions.

   Status: Fixed

Informational

2. Unnecessary approval

Line	Code

266	if (_stake) {
	//mint here and send to rewards on user behalf
	ITokenMinter(token).mint(address(this), _amount);
	address rewardContract = pool.pickleRewards;
	IERC20(token).safeApprove(rewardContract, 0);
	IERC20(token).safeApprove(rewardContract, _amount);
	IRewards(rewardContract).stakeFor(msg.sender, _amount);
	}

Description

In the deposit function, approval to rewardContract is set to first 0 and then the required amount. This pattern is appropriate in the normal approval. But when giving approval via another contract, then this is not needed, as there is no way to interrupt the current transaction.

Remediation

We recommend removing the line which sets the allowance first to zero.

Status: Fixed

3. Unnecessary variable assignment

Line	Code

68, 69, 70	constructor(address _staker, address _minter) public {
	isShutdown = false;
	staker = _staker;
	owner = msg.sender;
	voteDelegate = msg.sender;
	feeManager = msg.sender;
	poolManager = msg.sender;
	feeDistro = address(0); //
	address(0xA464e6DCda8AC41e03616F95f4BC98a13b8922Dc);
	feeToken = address(0); //
	address(0x6c3F90f043a72FA612cbac8115EE7e52BDe6E490);
	treasury = address(0);
	minter = _minter;
	}

Description
The variables feeDistro, feeToken, and treasury have empty assignments in the constructor. There is no need to assign address(0) assignment. Because by default, those variables hold address(0) value.

Remediation
We recommend removing those three value assignments.

Status: Fixed

4. **Consider using latest solidity version**

   pragma solidity 0.6.12;

   Description
   We recommend using the latest solidity version as there are so many breaking changes, and security optimization has been implemented in the latest solidity version.

   Remediation
   Consider using solidity version 0.8.7, which is the latest at the time of this audit.

   Status: Fixed

C. Contract - PickleDepositor.sol

High severity issues

No issues were found.

Medium severity issues

Low level severity issues

1. Lack of event emissions

Description

It is recommended to emit an event whenever there is a state changing function. This is useful to the clients to track the actions appropriately. Following functions should emit an event.

- setFeeManager
- setFees
- initialLock
- _lockPickle
- deposit

Remediation

We suggest adding events in above functions.

Status: Fixed

Informational

2. Unnecessary approval

Line	Code

140	if (depositOnly) {
	//mint for msg.sender
	ITokenMinter(minter).mint(msg.sender, _amount);
	} else {
	//mint here
	ITokenMinter(minter).mint(address(this), _amount);
	//stake for msg.sender
	IERC20(minter).safeApprove(_stakeAddress, 0);
	IERC20(minter).safeApprove(_stakeAddress, _amount);
	IRewards(_stakeAddress).stakeFor(msg.sender, _amount);
	}

Description

In the deposit function, approval to _stakeAddress is set to first 0 and then required amount. This pattern is appropriate in the normal approval. But when giving approval via another contract, then this is not needed, as there is no way to interrupt the current transaction.

Remediation
We recommend removing the line which sets the allowance first to zero.

Status: Fixed

3. Consider using latest solidity version

pragma solidity 0.6.12;

Description
We recommend using the latest solidity version as there are so many breaking changes, and security optimization has been implemented in the latest solidity version.

Remediation
Consider using solidity version 0.8.7, which is the latest at the time of this audit.

Status: Fixed

Functional test

File: PickleBaseRewardPool.sol

Function Names	Testing results
constructor	Passed
totalSupply	Passed
balanceOf	Passed
extraRewardsLength	Passed
addExtraReward	Passed
clearExtraRewards	Passed
updateReward	Passed
lastTimeRewardApplicable	Passed
rewardPerToken	Passed
earned	Passed
stake	Infinite loop possibility
stakeAll	Passed
stakeFor	Infinite loop possibility
withdraw	Infinite loop possibility
withdrawAll	Passed
withdrawAndUnwrap	Infinite loop possibility
withdrawAllAndUnwrap	Passed
getReward	Infinite loop possibility
getReward	Passed

donate	Passed
queueNewRewards	Passed
notifyRewardAmount	Passed
getAPY	Passed

File: PickleBooster.sol

Function Names	Testing results
constructor	Passed
setOwner	Passed
setFeeManager	Passed
setPoolManager	Passed
setFactories	Passed
setArbitrator	Passed
setVoteDelegate	Passed
setRewardContracts	Passed
setFeeInfo	Passed
setFees	Passed
setTreasury	Passed
poolLength	Passed
addPicklePool	Passed
shutdownPool	Passed
shutdownSystem	Passed

deposit	Passed
depositAll	Passed
withdraw	Passed
withdrawAll	Passed
withdrawTo	Passed
vote	Passed
voteGaugeWeight	Passed
_earmarkRewards	Passed
earmarkRewards	Passed
earmarkFees	Passed
rewardClaimed	Passed

File: PickleDepositor.sol

constructor	Passed
setFeeManager	Passed
setFees	Passed
initialLock	Passed
_lockPickle	Passed
lockPickle	Passed
deposit	Passed
deposit	Passed
depositAll	Passed

Automated Testing

Slither

Slither log >> PickleBaseRewardPool.sol

```
INFO:Detectors:
PickleBaseRewardPool.notifyRewardAmount(uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1168-1182) performs a multiplication on the result of a division:
        -rewardRate = reward.div(duration) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1171)
        -leftover = remaining.mul(rewardRate) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1174)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
PickleBaseRewardPool.rewardPerToken() (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#994-1004) uses a dangerous strict equality:
        - totalSupply() == 0 (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#995)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
PickleBaseRewardPool.queueNewRewards(uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1141-1166) should emit an event for:
        - queuedRewards = 0 (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1148)
        - queuedRewards = 0 (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1161)
        - queuedRewards = _rewards (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1163)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
PickleBaseRewardPool.constructor(uint256,address,address,address,address).operator_ (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#945) lacks a zero-check on :
                - operator = operator_ (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#951)
PickleBaseRewardPool.constructor(uint256,address,address,address,address).rewardManager_ (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#946) lacks a zero-check on :
                - rewardManager = rewardManager_ (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#952)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
PickleBaseRewardPool.stake(uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1014-1029) has external calls inside a loop: IRewards(extraRewards[i]).stake(msg.sender,_amount) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1019)
PickleBaseRewardPool.stakeFor(address,uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1037-1054) has external calls inside a loop: IRewards(extraRewards[i]).stake(_for,_amount) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1042)
PickleBaseRewardPool.withdraw(uint256,bool) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1056-1075) has external calls inside a loo
```

```
INFO:Detectors:
Reentrancy in PickleBaseRewardPool.donate(uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1136-1139):
        External calls:
        - IERC20(rewardToken).safeTransferFrom(msg.sender,address(this),_amount) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1137)
        State variables written after the call(s):
        - queuedRewards = queuedRewards.add(_amount) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1138)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in PickleBaseRewardPool.getReward(address,bool) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1109-1129):
        External calls:
        - rewardToken.safeTransfer(_account,reward) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1117)
        - IDeposit(operator).rewardClaimed(pid,_account,reward) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1118)
        Event emitted after the call(s):
        - RewardPaid(_account,reward) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1119)
Reentrancy in PickleBaseRewardPool.stake(uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1014-1029):
        External calls:
        - stakingToken.safeTransferFrom(msg.sender,address(this),_amount) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1025)
        Event emitted after the call(s):
        - Staked(msg.sender,_amount) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1026)
Reentrancy in PickleBaseRewardPool.stakeFor(address,uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1037-1054):
        External calls:
        - stakingToken.safeTransferFrom(msg.sender,address(this),_amount) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1050)
        Event emitted after the call(s):
        - Staked(_for,_amount) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1051)
Reentrancy in PickleBaseRewardPool.withdraw(uint256,bool) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1056-1075):
        External calls:
        - stakingToken.safeTransfer(msg.sender,amount) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1067)
        Event emitted after the call(s):
        - Withdrawn(msg.sender,amount) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1068)
Reentrancy in PickleBaseRewardPool.withdraw(uint256,bool) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1056-1075):
        External calls:
        - stakingToken.safeTransfer(msg.sender,amount) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1067)
        - getReward(msg.sender,true) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1071)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#869)
                - rewardToken.safeTransfer(_account,reward) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1117)
                - (success,returndata) = target.call{value: value}(data) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#180)
```

```
        - getReward(msg.sender,true) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1071)
                - (success,returndata) = target.call{value: value}(data) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#180)
        Event emitted after the call(s):
        - RewardPaid(_account,reward) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1119)
                - getReward(msg.sender,true) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1071)
Reentrancy in PickleBaseRewardPool.withdrawAndUnwrap(uint256,bool) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1081-1103):
        External calls:
        - IDeposit(operator).withdrawTo(pid,amount,msg.sender) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1095)
        Event emitted after the call(s):
        - Withdrawn(msg.sender,amount) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1096)
Reentrancy in PickleBaseRewardPool.withdrawAndUnwrap(uint256,bool) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1081-1103):
        External calls:
        - IDeposit(operator).withdrawTo(pid,amount,msg.sender) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1095)
        - getReward(msg.sender,true) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1100)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (../chetan/gaza/mycontracts/vetoken-rew
ardPool.sol#869)
                - rewardToken.safeTransfer(_account,reward) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1117)
                - (success,returndata) = target.call{value: value}(data) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#180)
                - IDeposit(operator).rewardClaimed(pid,_account,reward) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1118)
                - IRewards(extraRewards[i]).getReward(_account) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1125)
        External calls sending eth:
        - getReward(msg.sender,true) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1100)
                - (success,returndata) = target.call{value: value}(data) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#180)
        Event emitted after the call(s):
        - RewardPaid(_account,reward) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1119)
                - getReward(msg.sender,true) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1100)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
PickleBaseRewardPool.getReward(address,bool) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1109-1129) uses timestamp for comparisons
        Dangerous comparisons:
        - reward > 0 (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1115)
PickleBaseRewardPool.queueNewRewards(uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1141-1166) uses timestamp for comparison
s
        Dangerous comparisons:
        - block.timestamp >= periodFinish (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1146)
        - queuedRatio < newRewardRatio (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1159)
PickleBaseRewardPool.notifyRewardAmount(uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#1168-1182) uses timestamp for compari
```

```
INFO:Detectors:
Address.functionCall(address,bytes) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#128-130) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#157-163) is never used and should
 be removed
Address.functionDelegateCall(address,bytes) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#217-219) is never used and should be remov
ed
Address.functionDelegateCall(address,bytes,string) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#227-236) is never used and should b
ed
Address.functionDelegateCall(address,bytes,string) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#227-236) is never used and should b
e removed
Address.functionStaticCall(address,bytes) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#190-192) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#200-209) is never used and should be
removed
Address.sendValue(address,uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#103-108) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#832-845) is never used and should be rem
oved
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#847-854) is never used and sho
uld be removed
SafeMath.mod(uint256,uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#443-445) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#103-108):
        - (success) = recipient.call{value: amount}() (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#106)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#171-182)
:
        - (success,returndata) = target.call{value: value}(data) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#180)
Low level call in Address.functionStaticCall(address,bytes,string) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#200-209):
        - (success,returndata) = target.staticcall(data) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#207)
Low level call in Address.functionDelegateCall(address,bytes,string) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#227-236):
        - (success,returndata) = target.delegatecall(data) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function ICurveGauge.claim_rewards() (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#499) is not in mixedCase
Function ICurveGauge.reward_tokens(uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#501) is not in mixedCase
Function ICurveGauge.rewarded_token() (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#503) is not in mixedCase
Function ICurveVoteEscrow.create_lock(uint256,uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#507) is not in mixedCase
Function ICurveVoteEscrow.increase_amount(uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#509) is not in mixedCase
Function ICurveVoteEscrow.increase_unlock_time(uint256) (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#511) is not in mixedCase
Function ICurveVoteEscrow.smart_wallet_checker() (../chetan/gaza/mycontracts/vetoken-rewardPool.sol#515) is not in mixedCase
```

Slither log >> PickleBooster.sol

```
Last login: Sat Aug 28 17:16:22 2021 from 163.53.179.230
root@server:~# clear
root@server:~# slither "/chetan/gaza/mycontracts/PickleBooster.sol"
INFO:Detectors:
Reentrancy in PickleBooster.addPicklePool(address,address) (../chetan/gaza/mycontracts/PickleBooster.sol#1084-1109):
        External calls:
        - token = ITokenFactory(tokenFactory).CreateDepositToken(_lptoken) (../chetan/gaza/mycontracts/PickleBooster.sol#1092)
        - newRewardPool = IRewardFactory(rewardFactory).CreatePickleRewards(pid,token) (../chetan/gaza/mycontracts/PickleBooster.sol#1094
)
        State variables written after the call(s):
        - poolInfo.push(PoolInfo(_lptoken,token,_gauge,newRewardPool,false)) (../chetan/gaza/mycontracts/PickleBooster.sol#1097-1105)
Reentrancy in PickleBooster.setFeeInfo() (../chetan/gaza/mycontracts/PickleBooster.sol#1028-1042):
        External calls:
        - lockFees = IRewardFactory(rewardFactory).CreateTokenRewards(_feeToken,lockRewards,address(this)) (../chetan/gaza/mycontracts/Pi
ckleBooster.sol#1035-1039)
        State variables written after the call(s):
        - feeToken = _feeToken (../chetan/gaza/mycontracts/PickleBooster.sol#1040)
Reentrancy in PickleBooster.shutdownPool(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#1112-1122):
        External calls:
        - IStaker(staker).withdrawAll(pool.lptoken,pool.gauge) (../chetan/gaza/mycontracts/PickleBooster.sol#1117)
        State variables written after the call(s):
        - pool.shutdown = true (../chetan/gaza/mycontracts/PickleBooster.sol#1119)
Reentrancy in PickleBooster.shutdownSystem() (../chetan/gaza/mycontracts/PickleBooster.sol#1127-1143):
        External calls:
        - IStaker(staker).withdrawAll(token,gauge) (../chetan/gaza/mycontracts/PickleBooster.sol#1139-1141)
        State variables written after the call(s):
        - pool.shutdown = true (../chetan/gaza/mycontracts/PickleBooster.sol#1140)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
PickleBooster._earmarkRewards(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#1269-1313) ignores return value by IStaker(staker).c
laimPickle(gauge) (../chetan/gaza/mycontracts/PickleBooster.sol#1276)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
PickleBooster.setFees(uint256,uint256,uint256,uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#1044-1070) should emit an event for:

        - lockIncentive = _lockFees (../chetan/gaza/mycontracts/PickleBooster.sol#1065)
        - stakerIncentive = _stakerFees (../chetan/gaza/mycontracts/PickleBooster.sol#1066)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
PickleBooster.constructor(address,address)._staker (../chetan/gaza/mycontracts/PickleBooster.sol#964) lacks a zero-check on :
                - staker = _staker (../chetan/gaza/mycontracts/PickleBooster.sol#966)
PickleBooster.constructor(address,address)._minter (../chetan/gaza/mycontracts/PickleBooster.sol#964) lacks a zero-check on :
                - minter = _minter (../chetan/gaza/mycontracts/PickleBooster.sol#974)
PickleBooster.setOwner(address)._owner (../chetan/gaza/mycontracts/PickleBooster.sol#979) lacks a zero-check on :
                - owner = _owner (../chetan/gaza/mycontracts/PickleBooster.sol#981)
PickleBooster.setFeeManager(address)._feeM (../chetan/gaza/mycontracts/PickleBooster.sol#984) lacks a zero-check on :
                - feeManager = _feeM (../chetan/gaza/mycontracts/PickleBooster.sol#986)
PickleBooster.setPoolManager(address)._poolM (../chetan/gaza/mycontracts/PickleBooster.sol#989) lacks a zero-check on :
                - poolManager = _poolM (../chetan/gaza/mycontracts/PickleBooster.sol#991)
PickleBooster.setFactories(address,address)._rfactory (../chetan/gaza/mycontracts/PickleBooster.sol#994) lacks a zero-check on :
                - rewardFactory = _rfactory (../chetan/gaza/mycontracts/PickleBooster.sol#1001)
PickleBooster.setFactories(address,address)._tfactory (../chetan/gaza/mycontracts/PickleBooster.sol#994) lacks a zero-check on :
                - tokenFactory = _tfactory (../chetan/gaza/mycontracts/PickleBooster.sol#1002)
PickleBooster.setArbitrator(address)._arb (../chetan/gaza/mycontracts/PickleBooster.sol#1006) lacks a zero-check on :
                - rewardArbitrator = _arb (../chetan/gaza/mycontracts/PickleBooster.sol#1008)
PickleBooster.setVoteDelegate(address)._voteDelegate (../chetan/gaza/mycontracts/PickleBooster.sol#1011) lacks a zero-check on :
                - voteDelegate = _voteDelegate (../chetan/gaza/mycontracts/PickleBooster.sol#1013)
PickleBooster.setRewardContracts(address,address)._rewards (../chetan/gaza/mycontracts/PickleBooster.sol#1016) lacks a zero-check on :
                - lockRewards = _rewards (../chetan/gaza/mycontracts/PickleBooster.sol#1022)
PickleBooster.setRewardContracts(address,address)._stakerRewards (../chetan/gaza/mycontracts/PickleBooster.sol#1016) lacks a zero-check o
n :
                - stakerRewards = _stakerRewards (../chetan/gaza/mycontracts/PickleBooster.sol#1023)
PickleBooster.setTreasury(address)._treasury (../chetan/gaza/mycontracts/PickleBooster.sol#1072) lacks a zero-check on :
                - treasury = _treasury (../chetan/gaza/mycontracts/PickleBooster.sol#1074)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
PickleBooster.shutdownSystem() (../chetan/gaza/mycontracts/PickleBooster.sol#1127-1143) has external calls inside a loop: IStaker(staker)
.withdrawAll(token,gauge) (../chetan/gaza/mycontracts/PickleBooster.sol#1139-1141)
PickleBooster.voteGaugeWeight(address[],uint256[]) (../chetan/gaza/mycontracts/PickleBooster.sol#1256-1266) has external calls inside a l
oop: IStaker(staker).voteGaugeWeight(_gauge[i],_weight[i]) (../chetan/gaza/mycontracts/PickleBooster.sol#1263)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in PickleBooster.addPicklePool(address,address) (../chetan/gaza/mycontracts/PickleBooster.sol#1084-1109):
        External calls:
```

```
INFO:Detectors:
Reentrancy in PickleBooster.addPicklePool(address,address) (../chetan/gaza/mycontracts/PickleBooster.sol#1084-1109):
        External calls:
        - token = ITokenFactory(tokenFactory).CreateDepositToken(_lptoken) (../chetan/gaza/mycontracts/PickleBooster.sol#1092)
        - newRewardPool = IRewardFactory(rewardFactory).CreatePickleRewards(pid,token) (../chetan/gaza/mycontracts/PickleBooster.sol#1094
)
        State variables written after the call(s):
        - gaugeMap[_gauge] = true (../chetan/gaza/mycontracts/PickleBooster.sol#1106)
Reentrancy in PickleBooster.shutdownPool(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#1112-1122):
        External calls:
        - IStaker(staker).withdrawAll(pool.lptoken,pool.gauge) (../chetan/gaza/mycontracts/PickleBooster.sol#1117)
        State variables written after the call(s):
        - gaugeMap[pool.gauge] = false (../chetan/gaza/mycontracts/PickleBooster.sol#1120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in PickleBooster._withdraw(uint256,uint256,address,address) (../chetan/gaza/mycontracts/PickleBooster.sol#1190-1214):
        External calls:
        - ITokenMinter(token).burn(_from,_amount) (../chetan/gaza/mycontracts/PickleBooster.sol#1202)
        - IStaker(staker).withdraw(lptoken,gauge,_amount) (../chetan/gaza/mycontracts/PickleBooster.sol#1207)
        - IERC20(lptoken).safeTransfer(_to,_amount) (../chetan/gaza/mycontracts/PickleBooster.sol#1211)
        Event emitted after the call(s):
        - Withdrawn(_to,_pid,_amount) (../chetan/gaza/mycontracts/PickleBooster.sol#1213)
Reentrancy in PickleBooster.deposit(uint256,uint256,bool) (../chetan/gaza/mycontracts/PickleBooster.sol#1146-1179):
        External calls:
        - IERC20(lptoken).safeTransferFrom(msg.sender,staker,_amount) (../chetan/gaza/mycontracts/PickleBooster.sol#1157)
        - IStaker(staker).deposit(lptoken,gauge) (../chetan/gaza/mycontracts/PickleBooster.sol#1162)
        - ITokenMinter(token).mint(address(this),_amount) (../chetan/gaza/mycontracts/PickleBooster.sol#1167)
        - IERC20(token).safeApprove(rewardContract,0) (../chetan/gaza/mycontracts/PickleBooster.sol#1169)
        - IERC20(token).safeApprove(rewardContract,_amount) (../chetan/gaza/mycontracts/PickleBooster.sol#1170)
        - IRewards(rewardContract).stakeFor(msg.sender,_amount) (../chetan/gaza/mycontracts/PickleBooster.sol#1171)
        - ITokenMinter(token).mint(msg.sender,_amount) (../chetan/gaza/mycontracts/PickleBooster.sol#1174)
        Event emitted after the call(s):
        - Deposited(msg.sender,_pid,_amount) (../chetan/gaza/mycontracts/PickleBooster.sol#1177)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (../chetan/gaza/mycontracts/PickleBooster.sol#75-85) uses assembly
        - INLINE ASM (../chetan/gaza/mycontracts/PickleBooster.sol#81-83)
```

```
INFO:Detectors:
Address.functionCall(address,bytes) (../chetan/gaza/mycontracts/PickleBooster.sol#128-130) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#157-163) is never used and should be r
emoved
Address.functionDelegateCall(address,bytes) (../chetan/gaza/mycontracts/PickleBooster.sol#217-219) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (../chetan/gaza/mycontracts/PickleBooster.sol#227-236) is never used and should be rem
oved
Address.functionStaticCall(address,bytes) (../chetan/gaza/mycontracts/PickleBooster.sol#190-192) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (../chetan/gaza/mycontracts/PickleBooster.sol#200-209) is never used and should be remov
ed
Address.sendValue(address,uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#103-108) is never used and should be removed
MathUtil.min(uint256,uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#472-474) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#847-854) is never used and should b
e removed
SafeMath.mod(uint256,uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#443-445) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#103-108):
        - (success) = recipient.call{value: amount}() (../chetan/gaza/mycontracts/PickleBooster.sol#106)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (../chetan/gaza/mycontracts/PickleBooster.sol#171-182):
        - (success,returndata) = target.call{value: value}(data) (../chetan/gaza/mycontracts/PickleBooster.sol#180)
Low level call in Address.functionStaticCall(address,bytes,string) (../chetan/gaza/mycontracts/PickleBooster.sol#200-209):
        - (success,returndata) = target.staticcall(data) (../chetan/gaza/mycontracts/PickleBooster.sol#207)
Low level call in Address.functionDelegateCall(address,bytes,string) (../chetan/gaza/mycontracts/PickleBooster.sol#227-236):
        - (success,returndata) = target.delegatecall(data) (../chetan/gaza/mycontracts/PickleBooster.sol#234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function ICurveGauge.claim_rewards() (../chetan/gaza/mycontracts/PickleBooster.sol#499) is not in mixedCase
Function ICurveGauge.reward_tokens(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#501) is not in mixedCase
Function ICurveGauge.rewarded_token() (../chetan/gaza/mycontracts/PickleBooster.sol#503) is not in mixedCase
Function ICurveVoteEscrow.create_lock(uint256,uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#507) is not in mixedCase
Function ICurveVoteEscrow.increase_amount(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#509) is not in mixedCase
Function ICurveVoteEscrow.increase_unlock_time(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#511) is not in mixedCase
Function ICurveVoteEscrow.smart_wallet_checker() (../chetan/gaza/mycontracts/PickleBooster.sol#515) is not in mixedCase
Function IPickleVoteEscrow.create_lock(uint256,uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#531) is not in mixedCase
Function IPickleVoteEscrow.increase_amount(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#533) is not in mixedCase
Function IPickleVoteEscrow.increase_unlock_time(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#535) is not in mixedCase
```

```
Low level call in Address.functionDelegateCall(address,bytes,string) (../chetan/gaza/mycontracts/PickleBooster.sol#227-236):
        - (success,returndata) = target.delegatecall(data) (../chetan/gaza/mycontracts/PickleBooster.sol#234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function ICurveGauge.claim_rewards() (../chetan/gaza/mycontracts/PickleBooster.sol#499) is not in mixedCase
Function ICurveGauge.reward_tokens(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#501) is not in mixedCase
Function ICurveGauge.rewarded_token() (../chetan/gaza/mycontracts/PickleBooster.sol#503) is not in mixedCase
Function ICurveVoteEscrow.create_lock(uint256,uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#507) is not in mixedCase
Function ICurveVoteEscrow.increase_amount(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#509) is not in mixedCase
Function ICurveVoteEscrow.increase_unlock_time(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#511) is not in mixedCase
Function ICurveVoteEscrow.smart_wallet_checker() (../chetan/gaza/mycontracts/PickleBooster.sol#515) is not in mixedCase
Function IPickleVoteEscrow.create_lock(uint256,uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#531) is not in mixedCase
Function IPickleVoteEscrow.increase_amount(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#533) is not in mixedCase
Function IPickleVoteEscrow.increase_unlock_time(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#535) is not in mixedCase
Function IPickleVoteEscrow.smart_wallet_checker() (../chetan/gaza/mycontracts/PickleBooster.sol#539) is not in mixedCase
Function IVoting.vote_for_gauge_weights(address,uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#569) is not in mixedCase
Function IRegistry.get_registry() (../chetan/gaza/mycontracts/PickleBooster.sol#579) is not in mixedCase
Function IRegistry.get_address(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#581) is not in mixedCase
Function IRegistry.gauge_controller() (../chetan/gaza/mycontracts/PickleBooster.sol#583) is not in mixedCase
Function IRegistry.get_lp_token(address) (../chetan/gaza/mycontracts/PickleBooster.sol#585) is not in mixedCase
Function IRegistry.get_gauges(address) (../chetan/gaza/mycontracts/PickleBooster.sol#587) is not in mixedCase
Function IRewardFactory.CreateCrvRewards(uint256,address) (../chetan/gaza/mycontracts/PickleBooster.sol#739) is not in mixedCase
Function IRewardFactory.CreatePickleRewards(uint256,address) (../chetan/gaza/mycontracts/PickleBooster.sol#741) is not in mixedCase
Function IRewardFactory.CreateTokenRewards(address,address,address) (../chetan/gaza/mycontracts/PickleBooster.sol#743-747) is not in mixe
dCase
Function IStashFactory.CreateStash(uint256,address,address,uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#757-762) is not in mixe
dCase
Function ITokenFactory.CreateDepositToken(address) (../chetan/gaza/mycontracts/PickleBooster.sol#766) is not in mixedCase
Parameter PickleBooster.setOwner(address)._owner (../chetan/gaza/mycontracts/PickleBooster.sol#979) is not in mixedCase
Parameter PickleBooster.setFeeManager(address)._feeM (../chetan/gaza/mycontracts/PickleBooster.sol#984) is not in mixedCase
Parameter PickleBooster.setPoolManager(address)._poolM (../chetan/gaza/mycontracts/PickleBooster.sol#989) is not in mixedCase
Parameter PickleBooster.setFactories(address,address)._rfactory (../chetan/gaza/mycontracts/PickleBooster.sol#994) is not in mixedCase
Parameter PickleBooster.setFactories(address,address)._tfactory (../chetan/gaza/mycontracts/PickleBooster.sol#994) is not in mixedCase
Parameter PickleBooster.setArbitrator(address)._arb (../chetan/gaza/mycontracts/PickleBooster.sol#1006) is not in mixedCase
Parameter PickleBooster.setVoteDelegate(address)._voteDelegate (../chetan/gaza/mycontracts/PickleBooster.sol#1011) is not in mixedCase
Parameter PickleBooster.setRewardContracts(address,address)._rewards (../chetan/gaza/mycontracts/PickleBooster.sol#1016) is not in mixedC
ase
```

```
Parameter PickleBooster.withdraw(uint256,uint256)._pid (../chetan/gaza/mycontracts/PickleBooster.sol#1217) is not in mixedCase
Parameter PickleBooster.withdraw(uint256,uint256)._amount (../chetan/gaza/mycontracts/PickleBooster.sol#1217) is not in mixedCase
Parameter PickleBooster.withdrawAll(uint256)._pid (../chetan/gaza/mycontracts/PickleBooster.sol#1223) is not in mixedCase
Parameter PickleBooster.withdrawTo(uint256,uint256,address)._pid (../chetan/gaza/mycontracts/PickleBooster.sol#1232) is not in mixedCase
Parameter PickleBooster.withdrawTo(uint256,uint256,address)._amount (../chetan/gaza/mycontracts/PickleBooster.sol#1233) is not in mixedCa
se
Parameter PickleBooster.withdrawTo(uint256,uint256,address)._to (../chetan/gaza/mycontracts/PickleBooster.sol#1234) is not in mixedCase
Parameter PickleBooster.vote(uint256,address,bool)._voteId (../chetan/gaza/mycontracts/PickleBooster.sol#1245) is not in mixedCase
Parameter PickleBooster.vote(uint256,address,bool)._votingAddress (../chetan/gaza/mycontracts/PickleBooster.sol#1246) is not in mixedCase
Parameter PickleBooster.vote(uint256,address,bool)._support (../chetan/gaza/mycontracts/PickleBooster.sol#1247) is not in mixedCase
Parameter PickleBooster.voteGaugeWeight(address[],uint256[])._gauge (../chetan/gaza/mycontracts/PickleBooster.sol#1256) is not in mixedCa
se
Parameter PickleBooster.voteGaugeWeight(address[],uint256[])._weight (../chetan/gaza/mycontracts/PickleBooster.sol#1256) is not in mixedC
ase
Parameter PickleBooster.earmarkRewards(uint256)._pid (../chetan/gaza/mycontracts/PickleBooster.sol#1315) is not in mixedCase
Parameter PickleBooster.rewardClaimed(uint256,address,uint256)._pid (../chetan/gaza/mycontracts/PickleBooster.sol#1334) is not in mixedCa
se
Parameter PickleBooster.rewardClaimed(uint256,address,uint256)._address (../chetan/gaza/mycontracts/PickleBooster.sol#1335) is not in mix
edCase
Parameter PickleBooster.rewardClaimed(uint256,address,uint256)._amount (../chetan/gaza/mycontracts/PickleBooster.sol#1336) is not in mixe
dCase
Constant PickleBooster.pickle (../chetan/gaza/mycontracts/PickleBooster.sol#918) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PickleBooster.registry (../chetan/gaza/mycontracts/PickleBooster.sol#919) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PickleBooster.distributionAddressId (../chetan/gaza/mycontracts/PickleBooster.sol#920) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PickleBooster.voteOwnership (../chetan/gaza/mycontracts/PickleBooster.sol#921) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PickleBooster.voteParameter (../chetan/gaza/mycontracts/PickleBooster.sol#922) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PickleBooster.MaxFees (../chetan/gaza/mycontracts/PickleBooster.sol#928) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
PickleBooster.slitherConstructorConstantVariables() (../chetan/gaza/mycontracts/PickleBooster.sol#913-1347) uses literals with too many d
igits:
        - registry = address(0x0000000022D53366457F9d5E68Ec105046FC4383) (../chetan/gaza/mycontracts/PickleBooster.sol#919)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
withdrawAll(uint256) should be declared external:
        - PickleBooster.withdrawAll(uint256) (../chetan/gaza/mycontracts/PickleBooster.sol#1223-1228)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Slither log >> PickleDepositor.sol

```
INFO:Detectors:
Reentrancy in PickleDepositor.lockPickle() (../chetan/gaza/mycontracts/PickleDepositor.sol#993-1001):
        External calls:
        - _lockPickle() (../chetan/gaza/mycontracts/PickleDepositor.sol#994)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (../chetan/gaza/mycontracts/PickleDepos
itor.sol#869)
                - IERC20(pickle).safeTransfer(staker,pickleBalance) (../chetan/gaza/mycontracts/PickleDepositor.sol#971)
                - (success,returndata) = target.call{value: value}(data) (../chetan/gaza/mycontracts/PickleDepositor.sol#180)
                - IStaker(staker).increaseAmount(pickleBalanceStaker) (../chetan/gaza/mycontracts/PickleDepositor.sol#981)
                - IStaker(staker).increaseTime(unlockAt) (../chetan/gaza/mycontracts/PickleDepositor.sol#988)
        - ITokenMinter(minter).mint(msg.sender,incentivePickle) (../chetan/gaza/mycontracts/PickleDepositor.sol#998)
        External calls sending eth:
        - _lockPickle() (../chetan/gaza/mycontracts/PickleDepositor.sol#994)
                - (success,returndata) = target.call{value: value}(data) (../chetan/gaza/mycontracts/PickleDepositor.sol#180)
        State variables written after the call(s):
        - incentivePickle = 0 (../chetan/gaza/mycontracts/PickleDepositor.sol#999)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
PickleDepositor.initialLock() (../chetan/gaza/mycontracts/PickleDepositor.sol#950-965) performs a multiplication on the result of a divis
ion:
        -unlockInWeeks = (unlockAt / WEEK) * WEEK (../chetan/gaza/mycontracts/PickleDepositor.sol#956)
PickleDepositor._lockPickle() (../chetan/gaza/mycontracts/PickleDepositor.sol#968-991) performs a multiplication on the result of a divis
ion:
        -unlockInWeeks = (unlockAt / WEEK) * WEEK (../chetan/gaza/mycontracts/PickleDepositor.sol#984)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
PickleDepositor._lockPickle() (../chetan/gaza/mycontracts/PickleDepositor.sol#968-991) uses a dangerous strict equality:
        - pickleBalanceStaker == 0 (../chetan/gaza/mycontracts/PickleDepositor.sol#976)
PickleDepositor.initialLock() (../chetan/gaza/mycontracts/PickleDepositor.sol#950-965) uses a dangerous strict equality:
        - vepickle == 0 (../chetan/gaza/mycontracts/PickleDepositor.sol#954)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in PickleDepositor._lockPickle() (../chetan/gaza/mycontracts/PickleDepositor.sol#968-991):
        External calls:
        - IERC20(pickle).safeTransfer(staker,pickleBalance) (../chetan/gaza/mycontracts/PickleDepositor.sol#971)
        - IStaker(staker).increaseAmount(pickleBalanceStaker) (../chetan/gaza/mycontracts/PickleDepositor.sol#981)
```

```
INFO:Detectors:
PickleDepositor.setFees(uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#942-948) contains a tautology or contradiction:
        - _lockIncentive >= 0 && _lockIncentive <= 30 (../chetan/gaza/mycontracts/PickleDepositor.sol#945)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction
INFO:Detectors:
PickleDepositor.setFees(uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#942-948) should emit an event for:
        - lockIncentive = _lockIncentive (../chetan/gaza/mycontracts/PickleDepositor.sol#946)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
PickleDepositor.constructor(address,address)._staker (../chetan/gaza/mycontracts/PickleDepositor.sol#931) lacks a zero-check on :
                - staker = _staker (../chetan/gaza/mycontracts/PickleDepositor.sol#932)
PickleDepositor.constructor(address,address)._minter (../chetan/gaza/mycontracts/PickleDepositor.sol#931) lacks a zero-check on :
                - minter = _minter (../chetan/gaza/mycontracts/PickleDepositor.sol#933)
PickleDepositor.setFeeManager(address)._feeManager (../chetan/gaza/mycontracts/PickleDepositor.sol#937) lacks a zero-check on :
                - feeManager = _feeManager (../chetan/gaza/mycontracts/PickleDepositor.sol#939)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in PickleDepositor.deposit(uint256,bool,address) (../chetan/gaza/mycontracts/PickleDepositor.sol#1007-1046):
        External calls:
        - IERC20(pickle).safeTransferFrom(msg.sender,staker,_amount) (../chetan/gaza/mycontracts/PickleDepositor.sol#1016)
        - _lockPickle() (../chetan/gaza/mycontracts/PickleDepositor.sol#1017)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (../chetan/gaza/mycontracts/PickleDepos
itor.sol#869)
                - IERC20(pickle).safeTransfer(staker,pickleBalance) (../chetan/gaza/mycontracts/PickleDepositor.sol#971)
                - (success,returndata) = target.call{value: value}(data) (../chetan/gaza/mycontracts/PickleDepositor.sol#180)
                - IStaker(staker).increaseAmount(pickleBalanceStaker) (../chetan/gaza/mycontracts/PickleDepositor.sol#981)
                - IStaker(staker).increaseTime(unlockAt) (../chetan/gaza/mycontracts/PickleDepositor.sol#988)
        External calls sending eth:
        - _lockPickle() (../chetan/gaza/mycontracts/PickleDepositor.sol#1017)
                - (success,returndata) = target.call{value: value}(data) (../chetan/gaza/mycontracts/PickleDepositor.sol#180)
        State variables written after the call(s):
        - incentivePickle = 0 (../chetan/gaza/mycontracts/PickleDepositor.sol#1021)
Reentrancy in PickleDepositor.deposit(uint256,bool,address) (../chetan/gaza/mycontracts/PickleDepositor.sol#1007-1046):
        External calls:
        - IERC20(pickle).safeTransferFrom(msg.sender,address(this),_amount) (../chetan/gaza/mycontracts/PickleDepositor.sol#1025)
        State variables written after the call(s):
        - incentivePickle = incentivePickle.add(callIncentive) (../chetan/gaza/mycontracts/PickleDepositor.sol#1031)
```

```
INFO:Detectors:
PickleDepositor._lockPickle() (../chetan/gaza/mycontracts/PickleDepositor.sol#968-991) uses timestamp for comparisons
        Dangerous comparisons:
        - unlockInWeeks.sub(unlockTime) > 2 (../chetan/gaza/mycontracts/PickleDepositor.sol#987)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (../chetan/gaza/mycontracts/PickleDepositor.sol#75-85) uses assembly
        - INLINE ASM (../chetan/gaza/mycontracts/PickleDepositor.sol#81-83)
Address.verifyCallResult(bool,bytes,string) (../chetan/gaza/mycontracts/PickleDepositor.sol#244-264) uses assembly
        - INLINE ASM (../chetan/gaza/mycontracts/PickleDepositor.sol#256-259)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (../chetan/gaza/mycontracts/PickleDepositor.sol#128-130) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#157-163) is never used and should be
 removed
Address.functionDelegateCall(address,bytes) (../chetan/gaza/mycontracts/PickleDepositor.sol#217-219) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (../chetan/gaza/mycontracts/PickleDepositor.sol#227-236) is never used and should be r
emoved
Address.functionStaticCall(address,bytes) (../chetan/gaza/mycontracts/PickleDepositor.sol#190-192) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (../chetan/gaza/mycontracts/PickleDepositor.sol#200-209) is never used and should be rem
oved
Address.sendValue(address,uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#103-108) is never used and should be removed
MathUtil.min(uint256,uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#472-474) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#847-854) is never used and should
 be removed
SafeMath.mod(uint256,uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#443-445) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#103-108):
        - (success) = recipient.call{value: amount}() (../chetan/gaza/mycontracts/PickleDepositor.sol#106)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (../chetan/gaza/mycontracts/PickleDepositor.sol#171-182):
        - (success,returndata) = target.call{value: value}(data) (../chetan/gaza/mycontracts/PickleDepositor.sol#180)
Low level call in Address.functionStaticCall(address,bytes,string) (../chetan/gaza/mycontracts/PickleDepositor.sol#200-209):
        - (success,returndata) = target.staticcall(data) (../chetan/gaza/mycontracts/PickleDepositor.sol#207)
Low level call in Address.functionDelegateCall(address,bytes,string) (../chetan/gaza/mycontracts/PickleDepositor.sol#227-236):
        - (success,returndata) = target.delegatecall(data) (../chetan/gaza/mycontracts/PickleDepositor.sol#234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Function ICurveGauge.rewarded_token() (../chetan/gaza/mycontracts/PickleDepositor.sol#503) is not in mixedCase
Function ICurveVoteEscrow.create_lock(uint256,uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#507) is not in mixedCase
Function ICurveVoteEscrow.increase_amount(uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#509) is not in mixedCase
Function ICurveVoteEscrow.increase_unlock_time(uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#511) is not in mixedCase
Function ICurveVoteEscrow.smart_wallet_checker() (../chetan/gaza/mycontracts/PickleDepositor.sol#515) is not in mixedCase
Function IPickleVoteEscrow.create_lock(uint256,uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#531) is not in mixedCase
Function IPickleVoteEscrow.increase_amount(uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#533) is not in mixedCase
Function IPickleVoteEscrow.increase_unlock_time(uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#535) is not in mixedCase
Function IPickleVoteEscrow.smart_wallet_checker() (../chetan/gaza/mycontracts/PickleDepositor.sol#539) is not in mixedCase
Function IVoting.vote_for_gauge_weights(address,uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#569) is not in mixedCase
Function IRegistry.get_registry() (../chetan/gaza/mycontracts/PickleDepositor.sol#579) is not in mixedCase
Function IRegistry.get_address(uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#581) is not in mixedCase
Function IRegistry.gauge_controller() (../chetan/gaza/mycontracts/PickleDepositor.sol#583) is not in mixedCase
Function IRegistry.get_lp_token(address) (../chetan/gaza/mycontracts/PickleDepositor.sol#585) is not in mixedCase
Function IRegistry.get_gauges(address) (../chetan/gaza/mycontracts/PickleDepositor.sol#587) is not in mixedCase
Function IRewardFactory.CreateCrvRewards(uint256,address) (../chetan/gaza/mycontracts/PickleDepositor.sol#739) is not in mixedCase
Function IRewardFactory.CreatePickleRewards(uint256,address) (../chetan/gaza/mycontracts/PickleDepositor.sol#741) is not in mixedCase
Function IRewardFactory.CreateTokenRewards(address,address,address) (../chetan/gaza/mycontracts/PickleDepositor.sol#743-747) is not in mi
xedCase
Function IStashFactory.CreateStash(uint256,address,address,uint256) (../chetan/gaza/mycontracts/PickleDepositor.sol#757-762) is not in mi
xedCase
Function ITokenFactory.CreateDepositToken(address) (../chetan/gaza/mycontracts/PickleDepositor.sol#766) is not in mixedCase
Parameter PickleDepositor.setFeeManager(address)._feeManager (../chetan/gaza/mycontracts/PickleDepositor.sol#937) is not in mixedCase
Parameter PickleDepositor.setFees(uint256)._lockIncentive (../chetan/gaza/mycontracts/PickleDepositor.sol#942) is not in mixedCase
Parameter PickleDepositor.deposit(uint256,bool,address)._amount (../chetan/gaza/mycontracts/PickleDepositor.sol#1008) is not in mixedCase
Parameter PickleDepositor.deposit(uint256,bool,address)._lock (../chetan/gaza/mycontracts/PickleDepositor.sol#1009) is not in mixedCase
Parameter PickleDepositor.deposit(uint256,bool,address)._stakeAddress (../chetan/gaza/mycontracts/PickleDepositor.sol#1010) is not in mix
edCase
Parameter PickleDepositor.deposit(uint256,bool)._amount (../chetan/gaza/mycontracts/PickleDepositor.sol#1048) is not in mixedCase
Parameter PickleDepositor.deposit(uint256,bool)._lock (../chetan/gaza/mycontracts/PickleDepositor.sol#1048) is not in mixedCase
Parameter PickleDepositor.depositAll(bool,address)._lock (../chetan/gaza/mycontracts/PickleDepositor.sol#1052) is not in mixedCase
Parameter PickleDepositor.depositAll(bool,address)._stakeAddress (../chetan/gaza/mycontracts/PickleDepositor.sol#1052) is not in mixedCas
e
Constant PickleDepositor.pickle (../chetan/gaza/mycontracts/PickleDepositor.sol#917) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PickleDepositor.escrow (../chetan/gaza/mycontracts/PickleDepositor.sol#918) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Slither:/chetan/gaza/mycontracts/PickleDepositor.sol analyzed (28 contracts with 75 detectors), 64 result(s) found
```

Results

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

SOLIDITY STATIC ANALYSIS

PickleBaseRewardPool.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 171:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeApprove(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 832:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeIncreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 847:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in PickleBaseRewardPool.stake(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1014:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 256:16:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 991:28:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1146:12:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1153:30:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 106:27:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 180:50:

Low level calls:

Use of "delegatecall": should be avoided whenever possible.
External code, that is called can change the state of the calling contract and send ether from the caller's balance.
If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 234:50:

Gas costs:

Gas requirement of function PickleBaseRewardPool.clearExtraRewards is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 975:4:

Gas costs:

Gas requirement of function PickleBaseRewardPool.rewardPerToken is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 994:4:

Gas costs:

Gas requirement of function PickleBaseRewardPool.earned is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 1006:4:

Gas costs:

Gas requirement of function PickleBaseRewardPool.stake is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)

Delete dynamic array:

The "delete" operation when applied to a dynamically sized array in Solidity generates code to delete each of the elements contained. If the array is large, this operation can surpass the block gas limit and raise an OOG exception. Also nested dynamically sized objects can produce the same results.
more
Pos: 977:8:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point.
Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1018:8:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point.
Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1041:8:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point.
Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1060:8:

Constant/View/Pure functions:

Address.isContract(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 75:4:

Constant/View/Pure functions:

Address.functionStaticCall(address,bytes) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 190:4:

Constant/View/Pure functions:

Address.functionStaticCall(address,bytes,string) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 200:4:

Constant/View/Pure functions:

Address.verifyCallResult(bool,bytes,string) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 244:4:

Constant/View/Pure functions:

SafeERC20._callOptionalReturn(contract IERC20,bytes) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more

Similar variable names:

PickleBaseRewardPool.(uint256,address,address,address,address) : Variables have very similar names "rewardToken" and "rewardToken_". Note: Modifiers are currently not considered by this static analysis.
Pos: 950:8:

Similar variable names:

PickleBaseRewardPool.(uint256,address,address,address,address) : Variables have very similar names "rewardToken" and "rewardToken_". Note: Modifiers are currently not considered by this static analysis.
Pos: 950:29:

Similar variable names:

PickleBaseRewardPool.(uint256,address,address,address,address) : Variables have very similar names "stakingToken" and "stakingToken_". Note: Modifiers are currently not considered by this static analysis.
Pos: 949:8:

Similar variable names:

PickleBaseRewardPool.(uint256,address,address,address,address) : Variables have very similar names "stakingToken" and "stakingToken_". Note: Modifiers are currently not considered by this static analysis.
Pos: 949:30:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 104:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 107:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 177:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1057:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1142:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 428:15:

PickleBooster.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 171:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeApprove(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 832:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeIncreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 847:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 81:8:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 256:16:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.

more

Pos: 106:27:

Gas costs:

Gas requirement of function PickleBooster.staker is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 934:4:

Gas costs:

Gas requirement of function PickleBooster.minter is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 935:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1131:8:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1262:8:

Miscellaneous

Constant/View/Pure functions:

Address.isContract(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 75:4:

Constant/View/Pure functions:

Address.functionStaticCall(address,bytes) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 190:4:

Constant/View/Pure functions:

SafeERC20._callOptionalReturn(contract IERC20,bytes) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 864:4:

Similar variable names:

PickleBooster.setFactories(address,address) : Variables have very similar names "_rfactory" and "_tfactory". Note: Modifiers are currently not considered by this static analysis.
Pos: 1001:28:

Similar variable names:

PickleBooster.setFactories(address,address) : Variables have very similar names "_rfactory" and "_tfactory". Note: Modifiers are currently not considered by this static analysis.
Pos: 1002:27:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 205:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 232:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 488:8:

PickleBaseRewardPool.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 171:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeApprove(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 832:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeIncreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 847:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 81:8:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 256:16:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 955:31:

Low level calls:

Use of "delegatecall": should be avoided whenever possible.
External code, that is called can change the state of the calling contract and send ether from the caller's balance.
If this is wanted behaviour, use the Solidity library feature if possible.

more

Pos: 234:50:

Gas & Economy

Gas costs:

Gas requirement of function PickleDepositor.staker is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 926:4:

Constant/View/Pure functions:

Address.isContract(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 75:4:

Constant/View/Pure functions:

Address.functionStaticCall(address,bytes) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 190:4:

Similar variable names:

PickleDepositor.initialLock() : Variables have very similar names "pickle" and "vepickle". Note: Modifiers are currently not considered by this static analysis.

Pos: 954:12:

Similar variable names:

PickleDepositor.initialLock() : Variables have very similar names "pickle" and "vepickle". Note: Modifiers are currently not considered by this static analysis.

Pos: 961:49:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 104:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 107:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 1012:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 428:15:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 956:37:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 984:33:

SOLHINT LINTER

PickleBaseRewardPool.sol

```
 contracts/vetoken.sol/vetoken.sol:2:1: Error: Compiler version
0.6.12 does not satisfy the r semver requirement
contracts/vetoken.sol/vetoken.sol:81:9: Error: Avoid to use inline
assembly. It is acceptable only in rare cases
contracts/vetoken.sol/vetoken.sol:106:28: Error: Avoid to use low
level calls.
contracts/vetoken.sol/vetoken.sol:180:51: Error: Avoid to use low
level calls.
contracts/vetoken.sol/vetoken.sol:234:51: Error: Avoid to use low
level calls.
contracts/vetoken.sol/vetoken.sol:256:17: Error: Avoid to use inline
assembly. It is acceptable only in rare cases
contracts/vetoken.sol/vetoken.sol:499:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:501:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:503:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:507:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:509:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:511:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:515:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:531:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:533:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:535:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:539:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:569:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:579:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:581:5: Error: Function name must
```

```
be in mixedCase
contracts/vetoken.sol/vetoken.sol:583:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:585:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:587:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:739:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:741:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:743:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:757:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:766:5: Error: Function name must
be in mixedCase
contracts/vetoken.sol/vetoken.sol:907:1: Error: Contract has 17
states declarations but allowed no more than 15
contracts/vetoken.sol/vetoken.sol:913:29: Error: Constant name must
be in capitalized SNAKE_CASE
contracts/vetoken.sol/vetoken.sol:914:5: Error: Explicitly mark
visibility of state
contracts/vetoken.sol/vetoken.sol:915:5: Error: Explicitly mark
visibility of state
contracts/vetoken.sol/vetoken.sol:928:29: Error: Constant name must
be in capitalized SNAKE_CASE
contracts/vetoken.sol/vetoken.sol:991:29: Error: Avoid to make
time-based decisions in your business logic
contracts/vetoken.sol/vetoken.sol:1146:13: Error: Avoid to make
time-based decisions in your business logic
contracts/vetoken.sol/vetoken.sol:1153:31: Error: Avoid to make
time-based decisions in your business logic
contracts/vetoken.sol/vetoken.sol:1170:13: Error: Avoid to make
time-based decisions in your business logic
contracts/vetoken.sol/vetoken.sol:1173:50: Error: Avoid to make
time-based decisions in your business logic
contracts/vetoken.sol/vetoken.sol:1179:26: Error: Avoid to make
time-based decisions in your business logic
contracts/vetoken.sol/vetoken.sol:1180:24: Error: Avoid to make
time-based decisions in your business logic
```

PickleBooster.sol

```
contracts/vetoken.sol/PickleBooster.sol:2:1: Error: Compiler version
0.6.12 does not satisfy the r semver requirement
contracts/vetoken.sol/PickleBooster.sol:81:9: Error: Avoid to use
inline assembly. It is acceptable only in rare cases
contracts/vetoken.sol/PickleBooster.sol:106:28: Error: Avoid to use
low level calls.
contracts/vetoken.sol/PickleBooster.sol:180:51: Error: Avoid to use
low level calls.
contracts/vetoken.sol/PickleBooster.sol:234:51: Error: Avoid to use
low level calls.
contracts/vetoken.sol/PickleBooster.sol:256:17: Error: Avoid to use
inline assembly. It is acceptable only in rare cases
contracts/vetoken.sol/PickleBooster.sol:499:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:501:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:503:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:507:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:509:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:511:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:515:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:531:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:533:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:535:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:539:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:569:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:579:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:581:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:583:5: Error: Function name
must be in mixedCase
```

```
contracts/vetoken.sol/PickleBooster.sol:585:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:587:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:739:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:741:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:743:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:757:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:766:5: Error: Function name
must be in mixedCase
contracts/vetoken.sol/PickleBooster.sol:913:1: Error: Contract has
20 states declarations but allowed no more than 15
contracts/vetoken.sol/PickleBooster.sol:918:29: Error: Constant name
must be in capitalized SNAKE_CASE
contracts/vetoken.sol/PickleBooster.sol:919:29: Error: Constant name
must be in capitalized SNAKE_CASE
contracts/vetoken.sol/PickleBooster.sol:920:29: Error: Constant name
must be in capitalized SNAKE_CASE
contracts/vetoken.sol/PickleBooster.sol:921:29: Error: Constant name
must be in capitalized SNAKE_CASE
contracts/vetoken.sol/PickleBooster.sol:922:29: Error: Constant name
must be in capitalized SNAKE_CASE
contracts/vetoken.sol/PickleBooster.sol:928:29: Error: Constant name
must be in capitalized SNAKE_CASE
contracts/vetoken.sol/PickleBooster.sol:1117:67: Error: Code
contains empty blocks
contracts/vetoken.sol/PickleBooster.sol:1117:76: Error: Code
contains empty blocks
contracts/vetoken.sol/PickleBooster.sol:1141:21: Error: Code
contains empty blocks
```

PickleDepositor.sol

```
contracts/vetoken.sol/artifacts/PickleDepositor.sol:2:1: Error:
Compiler version 0.6.12 does not satisfy the r semver requirement
contracts/vetoken.sol/artifacts/PickleDepositor.sol:81:9: Error:
Avoid to use inline assembly. It is acceptable only in rare cases
contracts/vetoken.sol/artifacts/PickleDepositor.sol:106:28: Error:
Avoid to use low level calls.
contracts/vetoken.sol/artifacts/PickleDepositor.sol:180:51: Error:
Avoid to use low level calls.
contracts/vetoken.sol/artifacts/PickleDepositor.sol:234:51: Error:
Avoid to use low level calls.
contracts/vetoken.sol/artifacts/PickleDepositor.sol:256:17: Error:
Avoid to use inline assembly. It is acceptable only in rare cases
contracts/vetoken.sol/artifacts/PickleDepositor.sol:499:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:501:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:503:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:507:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:509:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:511:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:515:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:531:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:533:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:535:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:539:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:569:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:579:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:581:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:583:5: Error:
Function name must be in mixedCase
```

```
contracts/vetoken.sol/artifacts/PickleDepositor.sol:585:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:587:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:739:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:741:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:743:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:757:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:766:5: Error:
Function name must be in mixedCase
contracts/vetoken.sol/artifacts/PickleDepositor.sol:917:29: Error:
Constant name must be in capitalized SNAKE_CASE
contracts/vetoken.sol/artifacts/PickleDepositor.sol:918:29: Error:
Constant name must be in capitalized SNAKE_CASE
contracts/vetoken.sol/artifacts/PickleDepositor.sol:955:32: Error:
Avoid to make time-based decisions in your business logic
contracts/vetoken.sol/artifacts/PickleDepositor.sol:983:28: Error:
Avoid to make time-based decisions in your business logic
```

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the veToken platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the veToken team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Integer Overflow and Underflow vulnerabilities or Back-Door Entry were found in the contract, but relying on other contracts might cause Reentrancy Vulnerability.

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com