QuillAudits

Audit Report
September, 2021

LET'S GO!

Contents

Scope of Audit

The scope of this audit was to analyze and document the Ego Pool Protocol 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	0	3
Closed	2	0	7	6

Introduction

During the period of **August 23,2021 to August 31, 2021** - QuillAudits Team performed a security audit for the EgoPool Protocol smart contract.

The code for the audit was taken from following the official link:
Git Repo link: https://michaelshm@bitbucket.org/exilion/ego-contract.git
Git Branch: master

Note	Date	Commit Hash
Version 1	25/08/2021	2ec231bb242af539a247d9bc860b5efa0e2c9074
Version 2	03/09/2021	2d4818c57429819ca06191ecd4d0dcc387293c96

Issues Found – Code Review / Manual Testing

A. Contract - Base Pool

High severity issues

1. **Missing important setting for contract**

   Description
   RewardVault has not been set by the constructor and is used in many functions.

   Remediation
   It should either be set in the constructor or need some validation before using it in any function.

   Status: Fixed
   This issue was reported in Version 1 and found fixed in Version 2.

2. **Validate address**

   Description
   From withrawalRewards, owner can send egoToken to any address without checking if that address has staked or used as celebrity.

   Remediation
   If it is part of the plan, then please disregard it.

   Status: Fixed
   This issue was reported in Version 1 and found fixed in Version 2.

Medium severity issues

No issues were found.

Low level severity issues

3. Function accessibility

Description
deployRewardVault can be called only once and can be called by any user. stakerClaim, celebrityClaim can be called by any user instead of the staker/celebrity only.

Remediation
If it is part of the plan, then please disregard it.

Status: Fixed
This issue was reported in Version 1 and found fixed in Version 2.

4. Function input parameters lack of check

Description
Variable validation is not performed in the functions below : constructor, stake, celebrityClaim, setPlatformFeeAddress, setPlatformFee, setDecimalAdjustment, withrawalRewards, stakerClaim, emergencyClaim. In stake function, caller can set his own address as celebrity.

Remediation
There should be some variable that should not be address(0) or greater than 0, and if set percentage, then check for limit/range. For celebrityClaim, the entered address should be checked whether it is a celebrity address or not. In stakerClaim function, staker should be validated if he has staked any token or not. In the stake function, the caller can not use his own address as celebrity. In the emergencyClaim function, the entered address should be checked whether it is a celebrity address or not.

Status: Fixed
This issue was reported in Version 1 and found fixed in Version 2.

5. Missing error message

Description

Missing error message with require in below functions: deployRewardVault, stakerClaim, celebrityClaim, setPlatformFee.

Remediation

Add relevant error messages.

Status: Fixed

This issue was reported in Version 1 and found fixed in Version 2.

6. Critical operation lacks event log

Description

Event is missing for withrawalRewards.

Remediation

Event should be fired for withrawalRewards.

Status: Fixed

This issue was reported in Version 1 and found fixed in Version 2.

Informational

7. Spelling mistake

Line	Code
360	`function withrawalRewards(address to, uint256 amount) external onlyOwner{ _egoToken.transferFrom( address(RewardVault), to, amount ); }`

Description

Typing error in the function name.

Remediation

The function name should be withdrawalRewards.

Status: Fixed

This issue was reported in Version 1 and found fixed in Version 2.

B. Contract - Ego Pool

High severity issues

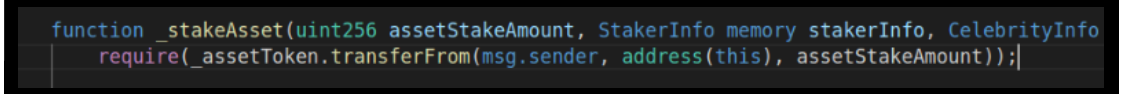
No issues were found.

Medium severity issues

No issues were found.

Low level severity issues

1. Missing error message

Line	Code
24	 `function _stakeAsset(uint256 assetStakeAmount, StakerInfo memory stakerInfo, CelebrityInfo` `require(_assetToken.transferFrom(msg.sender, address(this), assetStakeAmount));`

Description
When the required condition fails, there should be a message to show the error or cause of the failed transaction.

Remediation
Line no 24,40: add a message to require conditions.

Status: Fixed
This issue was reported in Version 1 and found fixed in Version 2.

Informational

2. Empty function used

Line	Code
35	`function _celebrityClaimAsset(address celebrity, CelebrityInfo memory celebrityInfo)  internal virtual override {` `}`

Description
_celebrityClaimAsset function has no code in it.

Remediation
Line no 35: _celebrityClaimAsset has no code in this function so remove this function.

Status: Acknowledged by the auditee
Auditee Comments: “This is required for inheritance. Can’t be removed.”

3. Unused parameters

Line	Code
24	`function _stakeAsset(address celebrity, uint256 assetStakeAmount, StakerInfo memory stakerInfo, CelebrityInfo memory celebrityInfo) internal virtual override{ if(assetStakeAmount > 0) require(_assetToken.transferFrom(msg.sender, address(this), assetStakeAmount)); uint256 vBalance = assetStakeAmount; stakerInfo.assetBalanceInfo.amount += assetStakeAmount; stakerInfo.assetBalanceInfo.vBalance += vBalance; }`
38	`function _unstakeAsset(address celebrity, StakerInfo memory stakerInfo, CelebrityInfo memory celebrityInfo, uint256 assetStakeAmount) internal virtual override { if(assetStakeAmount > 0) require(_assetToken.transfer(msg.sender, assetStakeAmount)); uint256 vBalance = assetStakeAmount; stakerInfo.assetBalanceInfo.amount -= assetStakeAmount; stakerInfo.assetBalanceInfo.vBalance -= vBalance; }`

Description
A celebrity address is not used in the function.

Remediation
Remove unused variables / parameters.

Status: Acknowledged by the auditee
Auditee Comments: “This is required for inheritance. Can’t be removed.”

C. Contract - Venus Pool

High severity issues

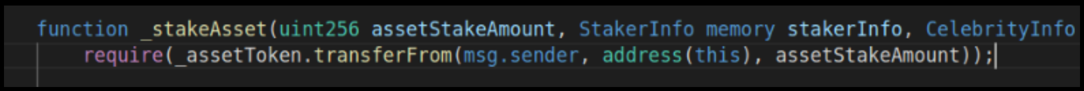
No issues were found.

Medium severity issues

No issues were found.

Low level severity issues

1. Missing error message

Line	Code
24	

```
function _stakeAsset(uint256 assetStakeAmount, StakerInfo memory stakerInfo, CelebrityInfo
    require(_assetToken.transferFrom(msg.sender, address(this), assetStakeAmount));
```

Description
When the required condition fails, there should be a message to show the error or cause of the failure of the transaction.

Remediation
Line no 31: add a message to require conditions.

Status: Fixed
This issue was reported in Version 1 and found fixed in Version 2.

Informational

2. Unused parameters

Description
Celebrity address is not used in the function.

Remediation
Remove unused parameters.

Status: Acknowledged by the auditee
Auditee Comments: “This is required for inheritance. Can’t be removed.”

3. Empty interface included

Description

Interface IEgoToken, there is no code in it.

Remediation

IBEP20.sol has been already imported into the current file, so no need of IEgoToken.sol

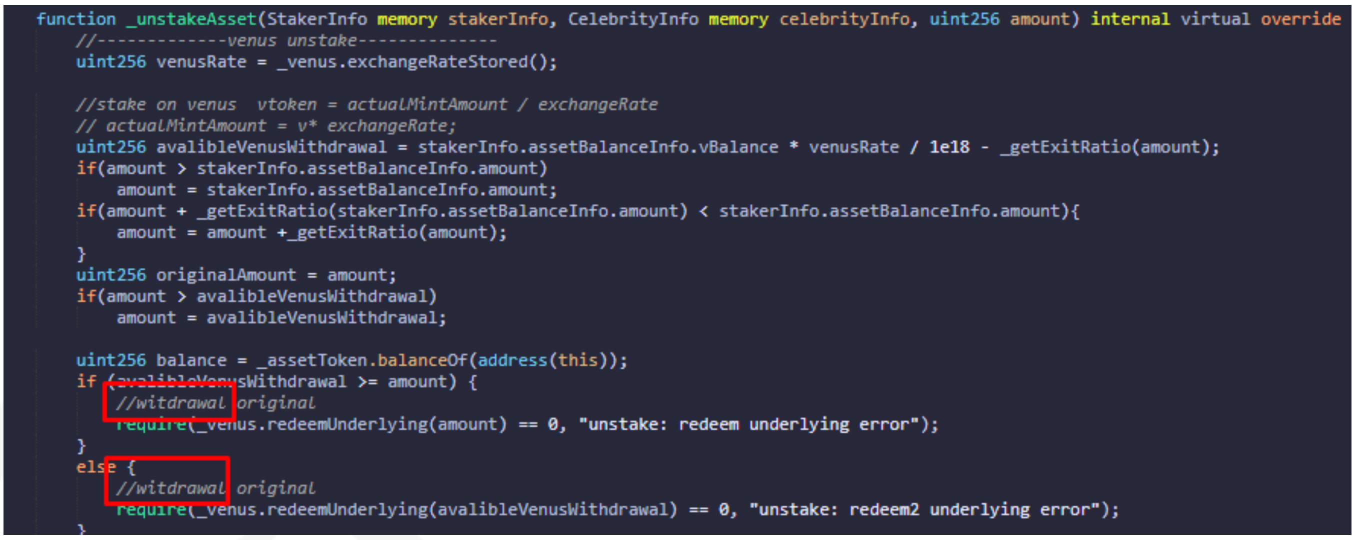
Status: Fixed

This issue was reported in Version 1 and found fixed in Version 2.

4. Spelling mistake

Description

Line:79 and 83, withdrawal word is written as witdrawal.



```
function _unstakeAsset(StakerInfo memory stakerInfo, CelebrityInfo memory celebrityInfo, uint256 amount) internal virtual override
    //------------venus unstake-------------
    uint256 venusRate = _venus.exchangeRateStored();

    //stake on venus  vtoken = actualMintAmount / exchangeRate
    // actualMintAmount = v* exchangeRate;
    uint256 avalibleVenusWithdrawal = stakerInfo.assetBalanceInfo.vBalance * venusRate / 1e18 - _getExitRatio(amount);
    if(amount > stakerInfo.assetBalanceInfo.amount)
        amount = stakerInfo.assetBalanceInfo.amount;
    if(amount + _getExitRatio(stakerInfo.assetBalanceInfo.amount) < stakerInfo.assetBalanceInfo.amount){
        amount = amount +_getExitRatio(amount);
    }
    uint256 originalAmount = amount;
    if(amount > avalibleVenusWithdrawal)
        amount = avalibleVenusWithdrawal;

    uint256 balance = _assetToken.balanceOf(address(this));
    if (avalibleVenusWithdrawal >= amount) {
        //witdrawal original
        require(_venus.redeemUnderlying(amount) == 0, "unstake: redeem underlying error");
    }
    else {
        //witdrawal original
        require(_venus.redeemUnderlying(avalibleVenusWithdrawal) == 0, "unstake: redeem2 underlying error");
    }
```

Remediation

Witdrawal word should be corrected by withdrawal.

Status: Fixed

This issue was reported in Version 1 and found fixed in Version 2.

D. Contract - MasterChef Pool

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low level severity issues

1. **Missing error message**

   Description
   When the required condition fails, there should be a message to show the error or cause of failure of the transaction.

   Remediation
   Line no 36: add a message to require conditions.

   Status: Fixed
   This issue was reported in Version 1 and found fixed in Version 2.

Informational

2. **Empty interface included**

   Description
   Interface IEgoToken, there is no code in it.

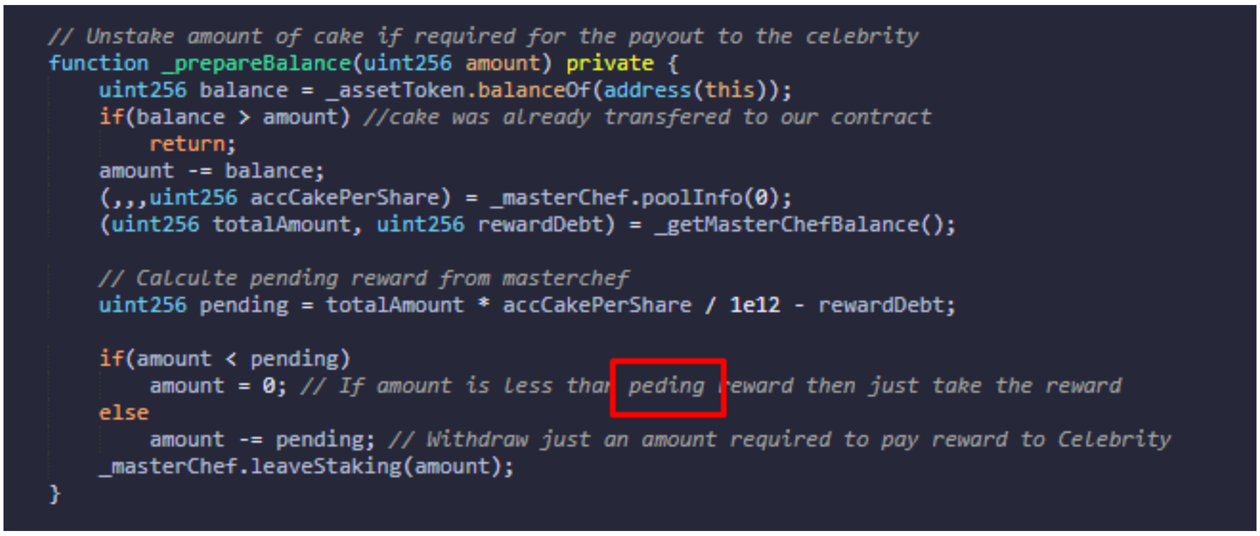
   Remediation
   IBEP20.sol has been already imported in the current file so no need of IEgoToken.sol

   Status: Fixed
   This issue was reported in Version 1 and found fixed in Version 2.

3. Spelling mistake

Description

Line:105, pending word is written as peding.



```
// Unstake amount of cake if required for the payout to the celebrity
function _prepareBalance(uint256 amount) private {
    uint256 balance = _assetToken.balanceOf(address(this));
    if(balance > amount) //cake was already transfered to our contract
        return;
    amount -= balance;
    (,,,uint256 accCakePerShare) = _masterChef.poolInfo(0);
    (uint256 totalAmount, uint256 rewardDebt) = _getMasterChefBalance();

    // Calculte pending reward from masterchef
    uint256 pending = totalAmount * accCakePerShare / 1e12 - rewardDebt;

    if(amount < pending)
        amount = 0; // If amount is less than peding reward then just take the reward
    else
        amount -= pending; // Withdraw just an amount required to pay reward to Celebrity
    _masterChef.leaveStaking(amount);
}
```

Remediation

peding word should be corrected by pending.

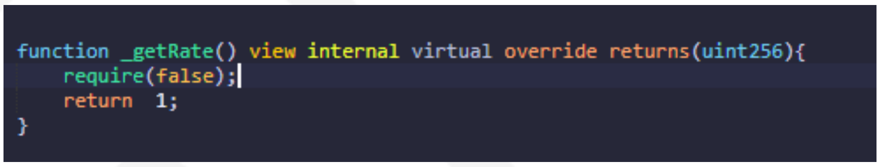
Status: Fixed

This issue was reported in Version 1 and found fixed in Version 2.

4. Invalid use of require

Description

require(false) will always revert.



```
function _getRate() view internal virtual override returns(uint256){
    require(false);
    return  1;
}
```

Remediation

Remove the code like this and return proper value.

Status: Fixed

This issue was reported in Version 1 and found fixed in Version 2.

Functional test

File: BasePool.sol

Function Names	Testing results
constructor	Passed
setEgoAssetRate	Passed
setDecimalAdjustment	Passed
deployRewardVault	Removed
stake	Passed
stakerClaim	Passed
celebrityClaim	Passed
calculateCelebrityEgoClaimAmount	Passed
calculateStakerEgoClaimAmount	Passed
_calculateClaimAmount	Passed
calculateCelebrityNetAssetReward	Passed
calculateCelebrityAssetClaimAmount	Passed
unstake	Passed
_updateRate	Passed
CalculatePoolRate	Passed
calculateAPR	Passed
_stakeAsset	Passed
_celebrityClaimAsset	Passed
_unstakeAsset	Passed

_getRate	Passed
setPlatformFee	Passed
setPlatformFeeAddress	Passed
withrawalRewards	Passed
owner	Passed
onlyOwner	Passed
renounceOwnership	Passed
transferOwnership	Passed

File: EgoPool.sol

Function Names	Testing results
constructor	Passed
_stakeAsset	Passed
_celebrityClaimAsset	Passed
_unstakeAsset	Passed
_getRate	Passed
owner	Passed
onlyOwner	Passed
renounceOwnership	Passed
transferOwnership	Passed

File: VenusPool.sol

constructor	Passed
_stakeAsset	Passed
_celebrityClaimAsset	Passed
_unstakeAsset	Passed
_getRate	Passed
_getExitRatio	Passed
owner	Passed
onlyOwner	Passed
renounceOwnership	Passed
transferOwnership	Passed

File: MasterChefPool.sol

Function Names	Testing results
constructor	Passed
_stakeAsset	Passed
_celebrityClaimAsset	Passed
_prepareBalance	Passed
_unstakeAsset	Passed
_getRate	Passed
owner	Passed
onlyOwner	Passed
renounceOwnership	Passed
transferOwnership	Passed
_msgSender	Passed
_msgData	Passed

File: PancakePriceProvider.sol

constructor	Passed
getPriceBase	Passed

File: Vault.sol

constructor	Passed

Automated Testing

Automated Test results can be found at the link below
https://docs.google.com/document/d/1yGwfWdzikZeyUowUcMU8m5HVsKqhoJ3dNatXfj1t7Qg/edit?usp=sharing

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the Ego Pool Protocol 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 Ego Pool 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. No instances of Integer Overflow and Underflow vulnerabilities or Back-Door Entry were found in the contract.

The audit showed several high, medium, low, and informational severity issues. We highly recommend addressing them before deploying the code.

The majority of the concerns addressed above have been acknowledged, implemented, and verified.

LET'S GO!

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com