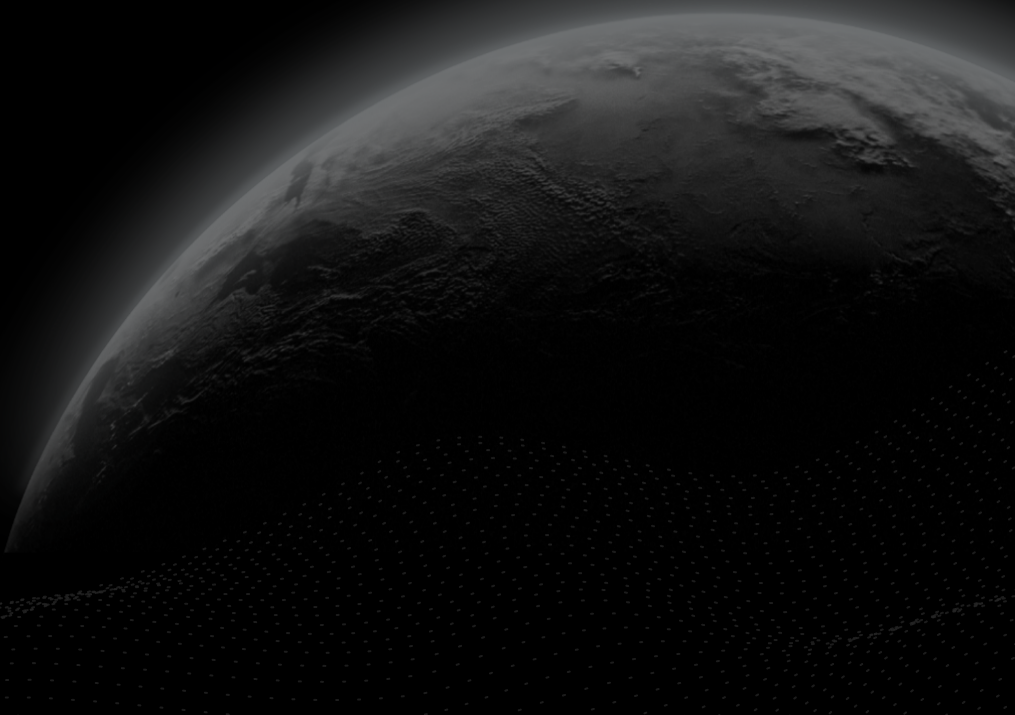




Security Assessment

New Trend Finance

CertiK Verified on Mar 27th, 2023





Certik Verified on Mar 27th, 2023

New Trend Finance

The security assessment was prepared by Certik, the leader in Web3.0 security.

Executive Summary

TYPES

DeFi, Staking

ECOSYSTEM

Ethereum (ETH) | Polygon

METHODS

Manual Review, Static Analysis

LANGUAGE

Solidity

TIMELINE

Delivered on 03/27/2023

KEY COMPONENTS

N/A

CODEBASE

<https://github.com/trendydefi/contracts/tree/452dea8a205d46f02f136048b5ef431966861144/Pools.sol>

[...View All](#)

COMMITTS

[452dea8a205d46f02f136048b5ef431966861144](https://github.com/trendydefi/contracts/tree/452dea8a205d46f02f136048b5ef431966861144)

[...View All](#)

Vulnerability Summary



19

Total Findings

11

Resolved

0

Mitigated

1

Partially Resolved

7

Acknowledged

0

Declined

0

Unresolved

0 Critical

Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.

2 Major

1 Resolved, 1 Partially Resolved



Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.

5 Medium

1 Resolved, 4 Acknowledged



Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.

7 Minor

6 Resolved, 1 Acknowledged



Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.

5 Informational

3 Resolved, 2 Acknowledged



Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | NEW TREND FINANCE

I **Summary**

Executive Summary

Vulnerability Summary

Codebase

Audit Scope

Approach & Methods

I **Decentralization Efforts**

Description

Recommendations

Short Term:

Long Term:

Permanent:

Alleviation

I **Third-Party Dependencies**

Description

Recommendations

I **Findings**

POO-01 : Lack of `userRank` check in `claimRankRewardMonthly()`

POO-02 : Not reasonable rank level system in the `logVolume()`

PCK-02 : Logical issue with vote and getStuck

PCK-03 : Lack of reward source

POO-03 : Logical issue in the `giveRankRewardMonthly()`

POO-04 : Rank reward winners may increase after the `giveRankRewardMonthly()` is invoked

PPB-01 : Logical issue about voting

PCK-05 : Missing Zero Address Validation

PCK-06 : Unchecked ERC-20 `transfer()`/`transferFrom()` Call

PCK-07 : Logical issue of `onlyCeo`

PCK-08 : `refer.userInfos` should be up-to-date

PCK-09 : Usage of `transfer`/`send` for sending Ether

POO-05 : Different elapsed time check

POO-08 : Potential incorrect decimal in `bnbPrice()`

PCK-10 : Missing Emit Events

PCK-11 : Declaration Naming Convention

PCK-12 : Redundant `referByParent`

POO-06 : Redundant `max`

POO-07 : Unnecessary `voteType` in `voteConfigs`

Appendix

Disclaimer

CODEBASE | NEW TREND FINANCE

Repository



<https://github.com/trendydefi/contracts/tree/452dea8a205d46f02f136048b5ef431966861144/Pools.sol>

Commit

[452dea8a205d46f02f136048b5ef431966861144](#)

AUDIT SCOPE | NEW TREND FINANCE

2 files audited ● 2 files with Acknowledged findings

ID	File	SHA256 Checksum
● PPB	 Pools.sol/Pools.sol	
● PCK	 Pools.sol	f0e15092f39b12622129732308f0ca48da6a07 800ebb3802d63e085b3ed83580

APPROACH & METHODS | NEW TREND FINANCE

This report has been prepared for New Trend Finance to discover issues and vulnerabilities in the source code of the New Trend Finance project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

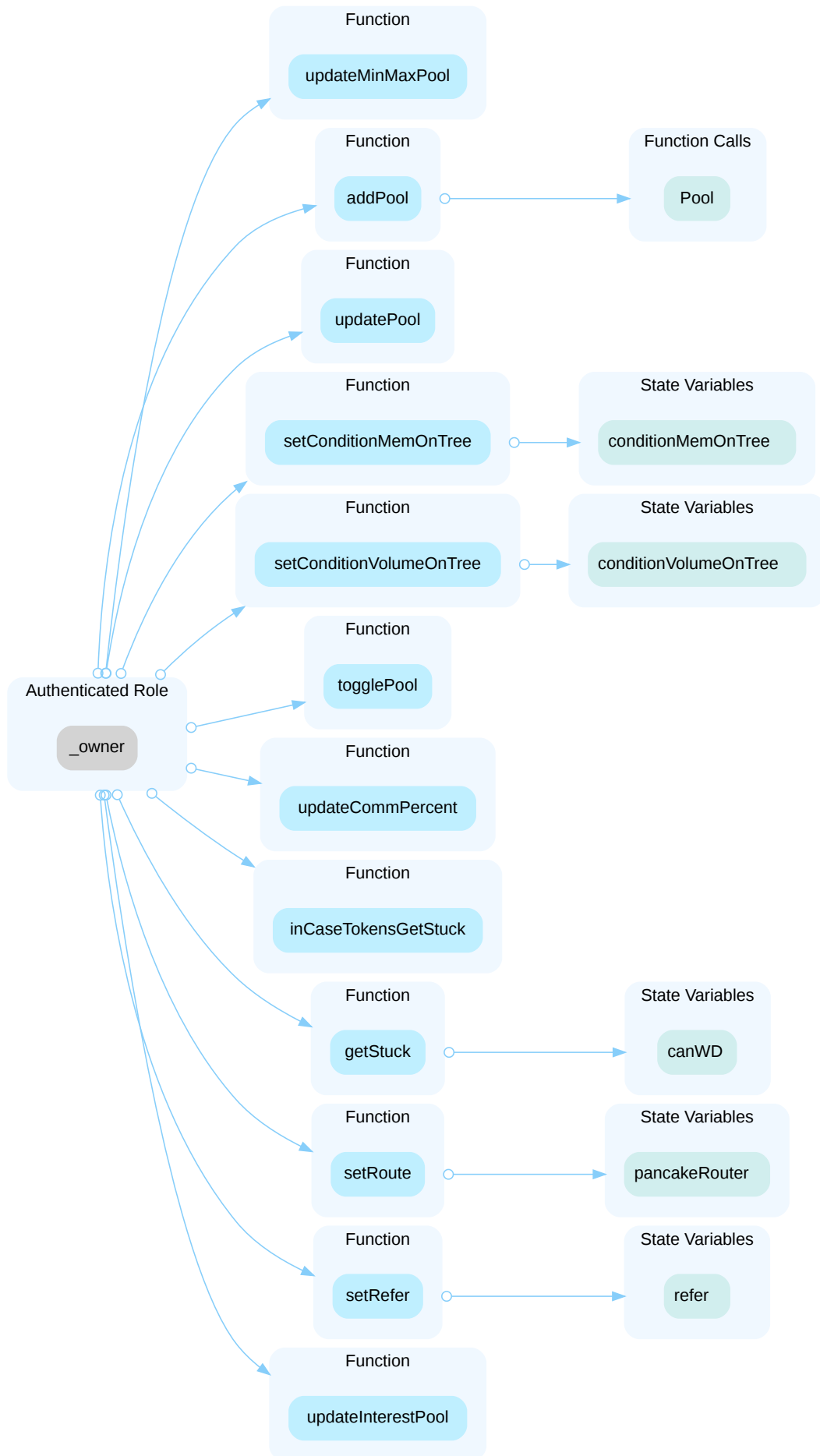
The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

DECENTRALIZATION EFFORTS | NEW TREND FINANCE

Description

In the contract `Pools` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



In the contract `Pools` the functions below can be called by `onlyCeo`.

- function `setCeo()`: to set a new `ceo`.
- function `adminRequestVote()`: to request a new round of votes.

Any compromise to the above-privileged account may allow the hacker to take advantage of this authority.

Recommendations

The risk describes the current project design and potentially makes iterations to improve the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term, and permanent:

Short Term:

Timelock and Multi sign ($\frac{2}{3}$, $\frac{3}{5}$) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness of privileged operations;
AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key being compromised;
AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness of privileged operations;
AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.

OR

- Remove the risky functionality.

I Alleviation

The team acknowledged this issue and decided to use the gnosisSafe multi-sig wallet for `ceo` in the future.

The team updated the code of the functions `updateInterestPool` and `updateCommPercent` in commit `1ff7c447136e997dc2a1cc204eee40948771e396`, which can only be called by the new modifier `onlyGnosisSafe`, and only if the vote to change these values has been approved. In addition, they plan to grant the role `gnosisSafe` to a multi-sign safe.

In addition, the team removed the `adminRequestVote()` and `getStuck()` in commit

`2355d0faa4eb84a1b6dc449173f691401c5a0a36`.

THIRD-PARTY DEPENDENCIES | NEW TREND FINANCE

Description

The contract is serving as the underlying entity to interact with one or more third-party protocols. The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties, migrating to new LP pools, etc.

```
28      IPancakeRouter public pancakeRouter;
```

- The contract `Pools` interacts with third party contract with `IPancakeRouter` interface via `pancakeRouter`.

```
29      IRefferal refer;
```

- The contract `Pools` interacts with third party contract with `IRefferal` interface via `refer`.

Recommendations

We recommend that the project team constantly monitor the functionality of these tools to mitigate any side effects that may occur when unexpected changes are introduced.

FINDINGS | NEW TREND FINANCE



19

Total Findings

0

Critical

2

Major

5

Medium

7

Minor

5

Informational

This report has been prepared to discover issues and vulnerabilities for New Trend Finance. Through this audit, we have uncovered 19 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
POO-01	Lack Of <code>userRank</code> Check In <code>claimRankRewardMonthly()</code>	Logical Issue	Major	Resolved
POO-02	Not Reasonable Rank Level System In The <code>logVolume()</code>	Logical Issue	Major	Partially Resolved
PCK-02	Logical Issue With Vote And GetStuck	Logical Issue	Medium	Resolved
PCK-03	Lack Of Reward Source	Logical Issue	Medium	Acknowledged
POO-03	Logical Issue In The <code>giveRankRewardMonthly()</code>	Logical Issue	Medium	Acknowledged
POO-04	Rank Reward Winners May Increase After The <code>giveRankRewardMonthly()</code> Is Invoked	Logical Issue	Medium	Acknowledged
PPB-01	Logical Issue About Voting	Logical Issue	Medium	Acknowledged
PCK-05	Missing Zero Address Validation	Volatile Code	Minor	Resolved
PCK-06	Unchecked ERC-20 <code>transfer()</code> / <code>transferFrom()</code> Call	Volatile Code	Minor	Resolved
PCK-07	Logical Issue Of <code>onlyCeo</code>	Logical Issue	Minor	Resolved
PCK-08	<code>refer.userInfos</code> Should Be Up-To-Date	Logical Issue	Minor	Resolved

ID	Title	Category	Severity	Status
PCK-09	Usage Of <code>transfer</code> / <code>send</code> For Sending Ether	Volatile Code	Minor	● Resolved
POO-05	Different Elapsed Time Check	Logical Issue	Minor	● Acknowledged
POO-08	Potential Incorrect Decimal In <code>bnbPrice()</code>	Logical Issue	Minor	● Resolved
PCK-10	Missing Emit Events	Coding Style	Informational	● Resolved
PCK-11	Declaration Naming Convention	Coding Style	Informational	● Resolved
PCK-12	Redundant <code>_referByParent</code>	Logical Issue	Informational	● Resolved
POO-06	Redundant <code>_max</code>	Logical Issue	Informational	● Acknowledged
POO-07	Unnecessary <code>voteType</code> In <code>voteConfigs</code>	Logical Issue	Informational	● Acknowledged

POO-01 | LACK OF `userRank` CHECK IN `claimRankRewardMonthly()`

Category	Severity	Location	Status
Logical Issue	● Major	Pools.sol (2355d0faa4eb84a1b6dc449173f691401c5a0a36): 328~334	● Resolved

Description

The `claimRankRewardMonthly()` function does not include a `userRank` check, which means that anyone can claim the highest level rank reward without meeting the required rank criteria.

Scenario

A user who is not on the `userRank` list could successfully call `claimRankRewardMonthly(5)` to receive the corresponding `rankRewards[5].rewardInMonth`. As a result, a malicious user could potentially use multiple accounts to drain the `rankRewards[rid].remainInMonth` for any level `rid` each month."

Recommendation

We advise the client to add the aforementioned check.

Alleviation

The team heeded our advise and resolved this issue in commit `a7c7884be53dd8942d4e74d7163e7a94dec46705`.

POO-02 | NOT REASONABLE RANK LEVEL SYSTEM IN THE `logVolume()`

Category	Severity	Location	Status
Logical Issue	● Major	Pools.sol (2355d0faa4eb84a1b6dc449173f691401c5a0a36): 23 6~267	● Partially Resolved

Description

In the `logVolume()`, a user's rank level is classified and upgraded by the `volumeOntree[_referBy]`, `childs[_referBy].downLine`, `userTotalLock[_referBy]`, and `childs[_referBy].direct`, however, the criteria is not reasonable and the corresponding code is incorrect.

Scenario

If a user has `volumeOntree[_referBy]` equal to 3500_000, `userTotalLock[_referBy]` equal to 60000, `childs[_referBy].direct` equal to 1, and `childs[_referBy].downLine` equal to 9, the user must wait until `childs[_referBy].direct >= 10 && childs[_referBy].downLine >= 500` to directly achieve rank level 5. Until the user achieves this, they cannot be on the rank reward list. Additionally, in this scenario, `rankRewards[4].totalMember` does not have to be reduced by 1 when the user achieves rank level 5.

Recommendation

We advise the client to modify the code as the aforementioned information.

Alleviation

The team has acknowledged this issue and made the necessary correction in commit

`db2edd5a3edd5b9908df648f660e0b502b32dac2`. Specifically, the team has corrected the logic to reduce the `totalMember` of `rankRewards` when upgrading, while leaving the rank level upgrade condition unchanged.

PCK-02 | LOGICAL ISSUE WITH VOTE AND GETSTUCK

Category	Severity	Location	Status
Logical Issue	● Medium	Pools.sol (new): 291	● Resolved

Description

If the vote is successful (i.e., if 30% of the locked assets agree), the `owner` can use the function **getStuck()** in the contract to transfer any number of BNBs to a particular EOA. If the vote is successful, there could not be enough BNBs in the contract for investors to be able to withdraw their assets or claim their rewards.

Recommendation

In order to safeguard the interests of the investors, we advise the client to set a maximum quantity of BNBs for each vote to withdraw and establish a higher threshold for votes to pass.

Alleviation

The team removed the related functions, that the `owner` cannot extract BNBs from the contract, and resolved this issue in commit `2355d0faa4eb84a1b6dc449173f691401c5a0a36`.

PCK-03 | LACK OF REWARD SOURCE

Category	Severity	Location	Status
Logical Issue	● Medium	Pools.sol (new): 2	● Acknowledged

Description

Investors can deposit BNBs to a pool in this project and withdraw them together with rewards after a predetermined lock-in period. They will also need to pay taxes, which will be transferred to the `ceo` address. Additionally, the referrers of any deposit will receive their own rewards. However, the source of these rewards is currently unknown, which means that later investors may not be able to withdraw their assets or claim their rewards.

Recommendation

We recommend that the client elaborate more on the source of the funds and ensure that there will always be sufficient funds available for withdrawal and reward.

Alleviation

The team acknowledged this issue and stated that they will transfer assets to the contract regularly to pay interest to users.

POO-03 | LOGICAL ISSUE IN THE `giveRankRewardMonthly()`

Category	Severity	Location	Status
Logical Issue	● Medium	Pools.sol (2355d0faa4eb84a1b6dc449173f691401c5a0a36): 278	● Acknowledged

Description

The `giveRankRewardMonthly()` function contains an `else` code block that sets `rankRewards[i].remainInMonth` and `rankRewards[i].rewardInMonth` to zero if `bnb2USD(rankRewards[i].total)` is less than `rankRewards[i].minStart`. This means that if the rank reward winners have not claimed their reward this month and the total rank reward is below the `minStart` threshold, their reward for this month will be zero.

Additionally, the `giveRankRewardTime` variable will be set to the current time, meaning that all the reward getters will have to wait for another month to receive their reward.

Recommendation

We advise the client to remove the `else` code block and move the `giveRankRewardTime = block.timestamp;` to the `if(bnb2USD(rankRewards[i].total) >= rankRewards[i].minStart)` code block.

Alleviation

The team acknowledged this issue and stated that this is by design and will be notified to all users, also they have mentioned that the `giveRankRewardTime` is not used to check claim reward, it is just making sure `admin` give a reward each ≥ 30 days.

POO-04 | RANK REWARD WINNERS MAY INCREASE AFTER THE `giveRankRewardMonthly()` IS INVOKED

Category	Severity	Location	Status
Logical Issue	● Medium	Pools.sol (2355d0faa4eb84a1b6dc449173f691401c5a0a36): 28 8	● Acknowledged

Description

After `giveRankRewardMonthly()` is executed, there is a possibility that the number of rank reward winners in each level could increase, leading to an insufficient `rankRewards[i].remainInMonth` for already calculated reward amount for users to claim.

Recommendation

We advise the client to consolidate the functionality of `giveRankRewardMonthly()` and `claimRankRewardMonthly()` into a single function.

Alleviation

The team acknowledged this issue and stated that:

"For the benefit of the referrers, we want to give them the opportunity to raise their level to join the higher pool, and every month if there is a reward, the user can only claim in 1 pool for 1 month. they can decide the strategy to decide whether to level up to receive the reward or keep holding the current level to receive the reward according to the current level."

PPB-01 | LOGICAL ISSUE ABOUT VOTING

Category	Severity	Location	Status
Logical Issue	● Medium	Pools.sol/Pools.sol (latest)	● Acknowledged

Description

There are several logical issues regarding vote-related codes:

1. The users who deposit and hold vote power are unaware of the proposed values for `currentInterest` and `commPercent` when the `gnosis` address initiates a vote by calling the `adminRequestVoteConfig()` function. Furthermore, the voting results are not reflected in the `updateInterestPool()` and `updateCommPercent()` methods, and the `currentInterest` and `commPercent` values can be arbitrarily modified by the `gnosis` address.
2. Once a voting round is completed and passed, its status will always be set to `2`. If the `gnosis` address does not initiate another voting cycle, the `currentInterest` and `commPercent` values can be changed at any time.
3. The `userVote` array in the `VoteConfig` seems to serve no purpose.

Recommendation

We advise the client to clearly define the values for `currentInterest` and `commPercent` during each vote round. In addition, rather than passing in external values in the setting functions, the results of the vote should be used to set the new values for `currentInterest` and `commPercent`. Finally, it is advised to make use of all components in the struct variable or just remove any redundant code.

Alleviation

The team acknowledged this issue and decided to leave it as it is for now.

PCK-05 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	Pools.sol (new): 82, 84, 85, 117, 219, 305	Resolved

Description

Addresses should be checked before assignment or external call to make sure they are not zero addresses.

```
82      ceo = _ceo;
```

- `_ceo` is not zero-checked before being used.

```
84      WBNB = _WBNBAddress;
```

- `_WBNBAddress` is not zero-checked before being used.

```
85      USD = _USDAddress;
```

- `_USDAddress` is not zero-checked before being used.

```
117     ceo = _ceo;
```

- `_ceo` is not zero-checked before being used.

```
219     to.transfer(remainComm[_msgSender()]);
```

- `to` is not zero-checked before being used.

```
305     user.transfer(amount);
```

- `user` is not zero-checked before being used.

Recommendation

We advise adding a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

The team heeded our advice and resolved this issue in commit `743ee413b8e67fb9f8a7e3607b8d9d07fccea1ed`.

PCK-06 | UNCHECKED ERC-20 `transfer()` / `transferFrom()` CALL

Category	Severity	Location	Status
Volatile Code	● Minor	Pools.sol (new): 285	● Resolved

Description

The return value of the `transfer()/transferFrom()` call is not checked.

```
285         _token.transfer(msg.sender, _amount);
```

Recommendation

Since some ERC-20 tokens return no values and others return a `bool` value, they should be handled with care. We advise using the [OpenZeppelin's SafeERC20.sol](#) implementation to interact with the `transfer()` and `transferFrom()` functions of external ERC-20 tokens. The OpenZeppelin implementation checks for the existence of a return value and reverts if `false` is returned, making it compatible with all ERC-20 token implementations.

Alleviation

The team heeded our advice and resolved this issue in commit `743ee413b8e67fb9f8a7e3607b8d9d07fccea1ed`.

PCK-07 | LOGICAL ISSUE OF `onlyCeo`

Category	Severity	Location	Status
Logical Issue	● Minor	Pools.sol (new): 76~79	● Resolved

Description

The `onlyCeo` modifier should verify whether the `msg.sender` is the `ceo`, rather than checking whether it is equal to `_owner`.

Recommendation

We advise the client to review the code and ensure the logical correctness.

Alleviation

The team heeded our advice and resolved this issue in commit `743ee413b8e67fb9f8a7e3607b8d9d07fccea1ed`.

PCK-08 | `refer.userInfos` SHOULD BE UP-TO-DATE

Category	Severity	Location	Status
Logical Issue	● Minor	Pools.sol (new): 234	● Resolved

I Description

It is important to verify that the `refer.userInfos(msg.sender)` has been updated before a user calls the `deposit()` function to ensure that the referring information is on record for the user.

I Recommendation

We advise the client to constantly monitor the status of refer and ensure it's up-to-date.

I Alleviation

The team acknowledged this issue and stated this is by design:

"In the referrer contract, we have already handled this logic via the register function, when registering, the user will belong to one of the referrers, we also allow the user to deposit without registering, in this case, will not reward referrers."

PCK-09 | USAGE OF `transfer` / `send` FOR SENDING ETHER

Category	Severity	Location	Status
Volatile Code	Minor	Pools.sol (new): 160, 174, 219~220	Resolved

Description

It is not recommended to use Solidity's `transfer()` and `send()` functions for transferring Ether, since some contracts may not be able to receive the funds. Those functions forward only a fixed amount of gas (2300 specifically) and the receiving contracts may run out of gas before finishing the transfer. Also, EVM instructions' gas costs may increase in the future. Thus, some contracts that can receive now may stop working in the future due to the gas limitation.

```
payable(_msgSender()).transfer(processAmount);
```

- `Pools.withdraw` uses `transfer()`.

```
payable(_msgSender()).transfer(processAmount);
```

- `Pools.claimReward` uses `transfer()`.

```
to.transfer(remainComm[_msgSender()]);
```

- `Pools.claimComm` uses `transfer()`.

Recommendation

We recommend using the `Address.sendValue()` function from OpenZeppelin.

Since `Address.sendValue()` may allow reentrancy, we also recommend guarding against reentrancy attacks by utilizing the Checks-Effects-Interactions Pattern or applying OpenZeppelin ReentrancyGuard.

Alleviation

The team heeded our advice and resolved this issue in commit `c04753a4830f0ae585d787a1c3485bd3d11772c3`.

POO-05 | DIFFERENT ELAPSED TIME CHECK

Category	Severity	Location	Status
Logical Issue	● Minor	Pools.sol (2355d0faa4eb84a1b6dc449173f691401c5a0a36): 274, 290	● Acknowledged

Description

The elapsed time check in `claimRankRewardMonthly()` and `giveRankRewardMonthly()` uses different time periods, which may cause discrepancies in counting the reward period. Specifically, `claimRankRewardMonthly()` uses `block.timestamp / getMonths()` to count the elapsed months since `block.timestamp` equals zero, while `giveRankRewardMonthly()` uses `block.timestamp - giveRankRewardTime > 30 days` to check if a month has passed since last `giveRankRewardTime`. This may cause confusion and inconsistencies in the reward system.

Recommendation

We advise the client to use the same elapsed time check.

Alleviation

The team acknowledged this issue and stated that:

"We have some rules here:

make sure the user does not claim on this month before the claim, if this month is not claimed, next month will reset, not plus total, make sure admin gives reward each ≥ 30 days, and need condition: total reward on pool $>$ min start then this month admin give 20% total reward on the pool to current month reward, then will have month not reward, not always have reward each month."

POO-08 | POTENTIAL INCORRECT DECIMAL IN `bnbPrice()`

Category	Severity	Location	Status
Logical Issue	● Minor	Pools.sol (2355d0faa4eb84a1b6dc449173f691401c5a0a36): 160~166	● Resolved

Description

When calling `IPancakeRouter(pancakeRouter).getAmountsIn()`, it's important to note that if the tokens in the path have different decimal values, the difference in decimals must be taken into account in order to calculate the correct `bnbPrice`. Specifically, the difference in decimals should be multiplied to obtain an accurate result.

Recommendation

We advise the client to specify the exact `chainId` and addresses for `usd` and `wBnb`, and multiply the difference in decimals if necessary.

Alleviation

The team heeded our advice and resolved this issue in

`https://polygonscan.com/address/0x849ACE7457cae40cd9B0e2C253bdb23357ecb523#code`.

PCK-10 | MISSING EMIT EVENTS

Category	Severity	Location	Status
Coding Style	● Informational	Pools.sol (new): 87, 90, 93, 113, 259, 262, 266, 269, 272, 280, 283, 303	● Resolved

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

The team heeded our advice and resolved this issue in commit `743ee413b8e67fb9f8a7e3607b8d9d07fccea1ed`.

PCK-11 | DECLARATION NAMING CONVENTION

Category	Severity	Location	Status
Coding Style	● Informational	Pools.sol (new): 33, 34, 87, 90, 93, 113, 116, 119, 130, 130, 280, 283	● Resolved

Description

One or more declarations do not conform to the [Solidity style guide](#) with regards to its naming convention.

Particularly:

- `camelCase` : Should be applied to function names, argument names, local and state variable names, modifiers
- `UPPER_CASE` : Should be applied to `constant` variables
- `CapWords` : Should be applied to contract names, struct names, event names, and enums

Recommendation

We recommend adjusting those variable and function names to properly conform to Solidity's naming convention.

Alleviation

The team heeded our advice and resolved this issue in commit `743ee413b8e67fb9f8a7e3607b8d9d07fccea1ed`.

PCK-12 | REDUNDANT `_referByParent`

Category	Severity	Location	Status
Logical Issue	● Informational	Pools.sol (new): 239	● Resolved

Description

The variable `_referByParent` is defined but not used anywhere, hence it is redundant.

Recommendation

We advise the client to remove the aforementioned codes.

Alleviation

The team heeded our advice and resolved this issue in commit `743ee413b8e67fb9f8a7e3607b8d9d07fccea1ed`.

POO-06 | REDUNDANT `_max`

Category	Severity	Location	Status
Logical Issue	● Informational	Pools.sol (2355d0faa4eb84a1b6dc449173f691401c5a0a36): 305	● Acknowledged

Description

The variable `_max` is calculated by not used to check anything, hence it is redundant.

Recommendation

We advise the client to remove the aforementioned codes.

Alleviation

The team acknowledged this issue and stated that:

"this variable is designed to let the user know the boundary of each level interest, but we also allow if the user deposit > max."

POO-07 | UNNECESSARY `voteType` IN `voteConfigs`

Category	Severity	Location	Status
Logical Issue	● Informational	Pools.sol (2355d0faa4eb84a1b6dc449173f691401c5a0a36): 455	● Acknowledged

Description

The `voteType` key in the `voteConfigs` map seems redundant since the `reqVote` already handles the same functionality as the `voteType`.

Recommendation

We advise the client to remove the redundant codes.

Alleviation

The team acknowledged this issue and stated that:

"We want to save history for each type request vote config comm/interest."

APPENDIX | NEW TREND FINANCE

Finding Categories

Categories	Description
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.



