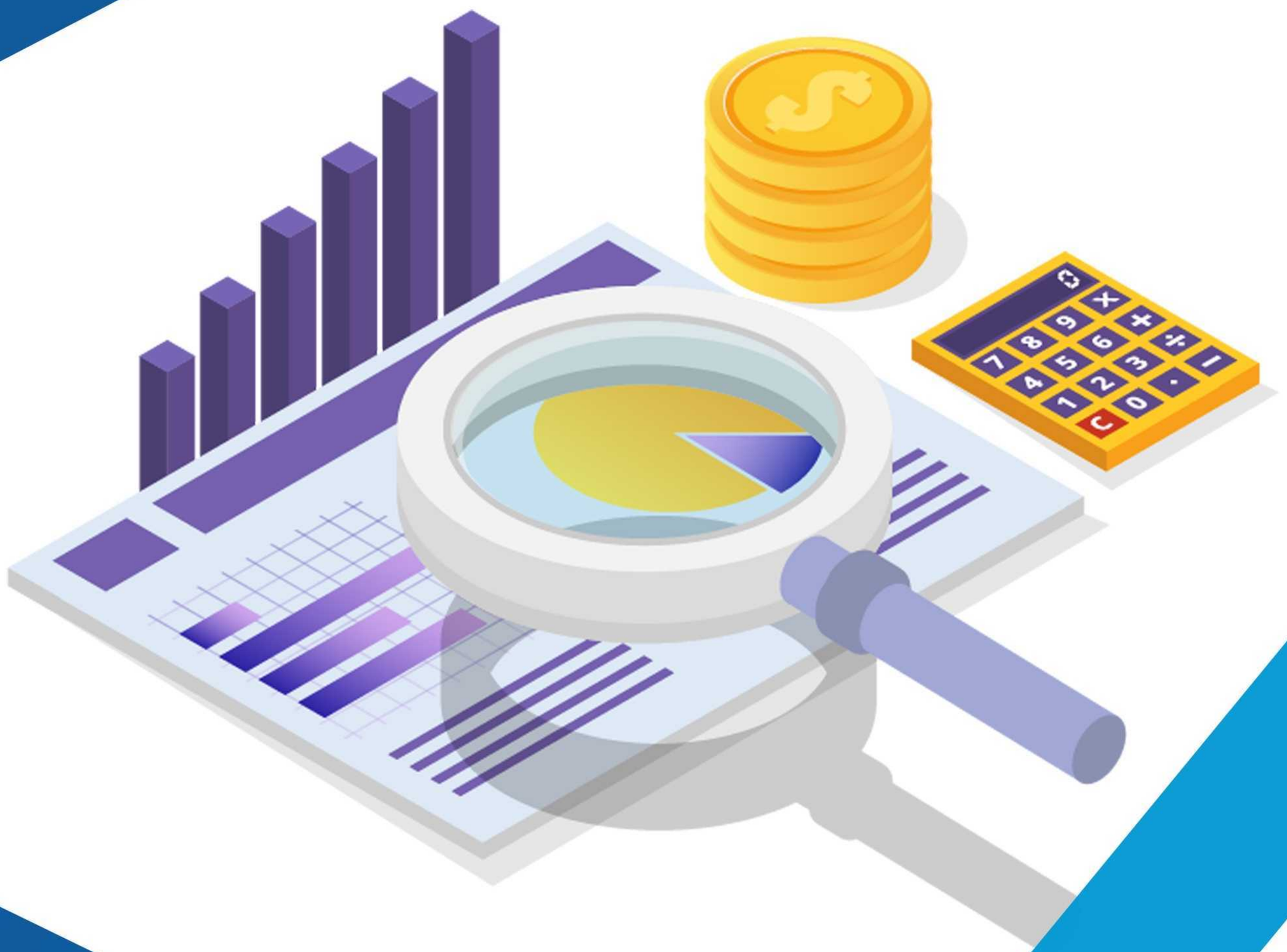
QuillAudits



Audit Report
July, 2021

Contents

Introduction

This Audit Report mainly focuses on the overall security of the Totem contracts. With this report, QuillAudits have tried to ensure the reliability and correctness of their smart contract by a complete and rigorous assessment of their system's architecture and the smart contract codebase.

Auditing Approach and Methodologies applied

The QuillAudits team has performed rigorous testing of the project, starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is well structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line-by-line inspection of the Smart Contract to find any potential issues like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was tested, and this included -

- Testing the functionality of the Smart Contract to determine proper logic has been followed throughout the whole process.
- Analyzing the complexity of the code in-depth and detailed, manual
- review of the code, line-by-line.
  Deploying the code on testnet using multiple clients to run live tests.
- Analyzing failure preparations to check how the Smart Contract performs in case of any bugs and vulnerabilities.
- Analyzing the security of the on-chain data.

Audit Details

Project Name: Totem
Code:
https://drive.google.com/file/d/1jPzoPpEIEthgDOB8P6jmfwMSogw_Ppgn/view?usp=sharing

Updated Code:
https://drive.google.com/file/d/1m-jrj8gKVY_PqXFwKJKmyJNDa12YsTc6/view?usp=sharing

Reviewed Code:
https://drive.google.com/file/d/1LCCuKAYRAnqsXWtc3mHdyWlgr-ug8uGR/view?usp=sharing

Languages: Solidity (Smart contract)
Platforms and Tools: Remix IDE, Metamask, Solhint, VScode, Contract Library, Slither.

Audit Goals

The focus of the audit was to verify that the Smart Contract System is secure, resilient and working according to the specifications. The audit activities can be grouped in the following three categories:

Security
Identifying security related issues within each contract and the system of contract.

Sound Architecture
Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.

Code Correctness and Quality
A full review of the contract source code. The primary areas of focus include:

- Accuracy
- Readability
- Sections of code with high complexity

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	1
Closed	3	1	5	4

Manual Audit

A. Contract: StakingPool.sol

High level severity issues

1. Calling a function inside the same function may cause the transaction to fail due to various reasons like the stack limit exceeding or the transaction not fitting into one block(because it consumes excessive gas). Inside the **_sortStakers()** function.

```
372         if (left < j) _sortStakers(left, j);
373         if (i < right) _sortStakers(i, right);
374     }
```

Status: Closed
The _sortStakers() function was taken off-chain.

2. Nested loops are not advisable in solidity due to the same reason mentioned above that the transaction might need so much gas that it does not fit inside a single bloc

```
339         while (i <= j) {
340             while (
341                 predictions[stakers[i].stakerAddress][stakers[i].index]
342                     .difference ==
343                 predictions[pivot.stakerAddress][pivot.index].difference
344                 ? predictions[stakers[i].stakerAddress][stakers[i].index]
345                     .stakedTime <
346                     predictions[pivot.stakerAddress][pivot.index].stakedTime
347                 : predictions[stakers[i].stakerAddress][stakers[i].index]
348                     .difference <
349                     predictions[pivot.stakerAddress][pivot.index].difference
350             ) i++;
351
352             while (
353                 predictions[pivot.stakerAddress][pivot.index].difference ==
354                     predictions[stakers[j].stakerAddress][stakers[j].index]
355                         .difference
356                 ? predictions[pivot.stakerAddress][pivot.index].stakedTime <
357                     predictions[stakers[j].stakerAddress][stakers[j].index]
358                         .stakedTime
359                 : predictions[pivot.stakerAddress][pivot.index].difference <
360                     predictions[stakers[j].stakerAddress][stakers[j].index]
361                         .difference
362             ) j--;
```

```
467             uint256 reward = 0;
468             uint256 btcReward = 0;
469             for (uint256 i = 0; i < userStakes.length; i++) {
470                 for (uint256 j = 0; j < prizeRewardRates.length; j++) {
471                     if (userStakes[i].rank <= prizeRewardRates[j].rank) {
472                         reward = reward.add(
473                             prizeAmount.mul(prizeRewardRates[j].percentage).div(
474                                 10000
475                             )
476                         );
477                         btcReward = btcReward.add(
478                             maturingBTCPrizeAmount
479                                 .mul(prizeRewardRates[j].percentage)
480                                 .div(10000)
481                         );
482                         break;
483                     }
484                 }
485             }
```

Status: Closed
All nested loops have been removed.

3. The logic to deduct the stakeTaxAmount is incorrect. If the stakeTaxRate > tokenTaxRate, then according to the logic, the stakeTaxAmount is deducted from the StakingPool's balance. This can lead to a loss for the StakingPool.

Status: Closed
stakeTaxAmount is deducted from the initial amount entered by the user now.

Medium severity issues

A. Contract: StakingPool.sol

1. getPrizeReward() function throws an error if called before ending the Pool. Due to this other functions using it, like getTotalReward() and claimReward() also throw the same error if called before ending the Pool. Consider adding a require(isMatured == true) in getPrizeReward().

Status: Closed
Implemented and verified.

Low level severity issues

A. Contract: StakingPool.sol

1. We don't need to import the StakingPoolFactory contract in the StakingPool contract because we're only saving the StakingPoolFactory address and not calling any of its functions. Instead, we can declare it like: **address public poolFactory;**

```
41        StakingPoolFactory public poolFactory;
```

Status: Closed
It was removed from the updated contract.

2. Many variables can be made immutable or constant to save gas, like sizeAllocation can be made immutable. Also, sizeLimitRangeRate can be made constant and initialized to 5.

Status: Closed
Implemented and verified.

3. There are multiple unnecessary read operations from storage.Like In _sortStakers() stakers.length is read at every loop. It is advised to store it in a variable and use that variable instead.

Status: Closed
Implemented and verified.

B. Contract: RewardManager.sol

1. The variable operator is a redundant variable as it is never initialized or updated

```
11        address operator;
```

Status: Acknowledged by the Auditee

C. Contract: StakingPoolFactory.sol

1. We don't need to import Initializable.sol in the StakingPoolFactory as it is already imported in the PoolCreator contract.

```
4   import "@openzeppelin/contracts/proxy/Initializable.sol";
5
6   import "./ContextUpgradeable.sol";
7   import "./Roles.sol";
8
9   contract PoolCreator is Initializable, ContextUpgradeable {
```

Status: Closed
It was removed from the updated contract.

D. Contract: PriceConsumer.sol

1. Price data received from the chainlink oracle is of type int256. While returning the data, it is typecasted to uint256. If the price received from the oracle is less than 0, it will result in wrong calculations in the StakingPool. Consider using a require check for (price > 0).

Status: Closed
Implemented and verified.

Informational

1. You can add a withdraw function to withdraw any ERC20 tokens stuck in the contract. There is one for WBTC, and USDC can be converted to WBTC and withdrawn, but instead of marginally losing tokens while swapping and paying for extra gas fees, it is better to add a withdraw function.

Status: Closed
Implemented and verified.

2. In the claimReward() function in the Withdraw event, btcReward can be included.

```
172     function claimReward() external {
173         (uint256 reward, uint256 btcReward) = _getTotalReward(_msgSender());
174
175         if (reward > 0) {
176             if (totemToken.balanceOf(address(rewardManager)) >= reward) {
177                 rewardManager.rewardUser(_msgSender(), reward);
178             }
179         }
180         if (btcReward > 0) require(btcToken.transfer(_msgSender(), btcReward));
181
182         _withdrawStakingReward(_msgSender());
183         _withdrawPrizeReward(_msgSender());
184
185         if (isMatured) {
186             uint256 stakedBalance = _getTotalStakedBalance(_msgSender());
187             if (stakedBalance > 0) {
188                 totemToken.transfer(_msgSender(), stakedBalance);
189                 _withdrawStakedBalance(_msgSender());
190
191                 emit Unstake(_msgSender(), stakedBalance);
192             }
193         }
194
195         emit Withdraw(_msgSender(), reward);
196     }
```

Status: Closed
btcReward was included in the event.

3. Maintain uniformity in _msgSender() use. There are multiple occurrences where msg.sender is used.

   Status: Closed
   Implemented and verified.

4. Consider using Minimal Proxy to reduce the costs of deploying StakingPool. More info: Minimal Proxy Contract

   Status: Acknowledged by the Auditee

5. In _sortStakers() initialize ‘uint256 i’ with 1, to avoid an unrequired iteration.

```
332     function _sortStakers() internal {
333         for (uint256 i = 0; i < stakers.length; i++) {
```

Status: Closed
_sortStakers function is calculated off-chain now.

Automated Testing

Solhint Linting Violations

Solhint is an open-source project for linting solidity code, providing both security and style guide validations. It integrates seamlessly into most mainstream IDEs. We used Solhint as a plugin within our VScode for this analysis. Multiple Linting violations were detected by Solhint; it is recommended to use Solhint's npm package to lint the contract.

Contract Library

Contract-library contains the most complete, high-level decompiled representation of all Ethereum smart contracts, with security analysis applied to these in real-time.
We performed analysis using the contract Library on the Ropsten address of the Totem contracts used during manual testing.
The contract library raised various warnings, which you can see in the respective analysis links.

Slither

Slither is an open-source static analysis framework. This tool provides rich information about Ethereum smart contracts and has critical properties. While Slither is built as a security-oriented static analysis framework, it is also used to enhance the user's understanding of smart contracts, assist in code reviews, and detect missing optimisations. Slither detected the following issues:

```
INFO:Detectors:
StakingPool.stake(uint256,uint256) (Staking\StakingPool.sol#120-168) ignores return value by totemToken.transferFrom(_msgSender(),addre
ss(this),_amount) (Staking\StakingPool.sol#136)
StakingPool.stake(uint256,uint256) (Staking\StakingPool.sol#120-168) ignores return value by totemToken.transfer(totemToken.taxationWal
let(),stakeTaxAmount) (Staking\StakingPool.sol#140)
StakingPool.claimReward() (Staking\StakingPool.sol#170-194) ignores return value by totemToken.transfer(_msgSender(),stakedBalance) (St
aking\StakingPool.sol#186)
StakingPool.releaseBTC(uint256,address) (Staking\StakingPool.sol#554-559) ignores return value by btcToken.transfer(receiver,amount) (S
taking\StakingPool.sol#558)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
PrivateSaleVesting.vested(address) (TotemVesting\PrivateSaleVesting.sol#30-82) performs a multiplication on the result of a division:
        -lockAmount = _vestingSchedule.totalAmount.sub(initialUnlockAmount).div(releasePeriods - 1) (TotemVesting\PrivateSaleVesting.so
l#72-75)
        -vestedAmount = period.sub(lockPeriods + 1).mul(lockAmount).add(initialUnlockAmount) (TotemVesting\PrivateSaleVesting.sol#77-80
)
StakingPool._getPrizeReward(address) (Staking\StakingPool.sol#429-468) performs a multiplication on the result of a division:
        -maturingBTCPrizeAmount = (usdPrizeAmount.mul(10 ** 10)).div(maturingPrice) (Staking\StakingPool.sol#435-436)
        -btcReward = btcReward.add(maturingBTCPrizeAmount.mul(prizeRewardRates[j].percentage).div(10000)) (Staking\StakingPool.sol#452-
456)
TotemVesting.vested(address) (TotemVesting\TotemVesting.sol#96-135) performs a multiplication on the result of a division:
        -lockAmount = _vestingSchedule.totalAmount.div(releasePeriods) (TotemVesting\TotemVesting.sol#131)
        -vestedAmount = period.sub(lockPeriods).mul(lockAmount) (TotemVesting\TotemVesting.sol#133)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
TotemVesting.vested(address) (TotemVesting\TotemVesting.sol#96-135) uses a dangerous strict equality:
        - ! isStartTimeSet || (_vestingSchedule.totalAmount == 0) || (lockPeriods == 0 && releasePeriods == 0) || (block.timestamp < st
artTime) (TotemVesting\TotemVesting.sol#104-107)
PrivateSaleVesting.vested(address) (TotemVesting\PrivateSaleVesting.sol#30-82) uses a dangerous strict equality:
        - ! isStartTimeSet || (_vestingSchedule.totalAmount == 0) || (lockPeriods == 0 && releasePeriods == 0) || (block.timestamp < st
artTime) (TotemVesting\PrivateSaleVesting.sol#38-41)
PrivateSaleVesting.vested(address) (TotemVesting\PrivateSaleVesting.sol#30-82) uses a dangerous strict equality:
        - period.sub(lockPeriods) == 1 (TotemVesting\PrivateSaleVesting.sol#68)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in TotemToken._transferWithTax(address,address,uint256) (TotemToken.sol#157-171):
        External calls:
        - _transfer(sender,taxationWallet,tax) (TotemToken.sol#169)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(sender,recipient,tokensToTransfer) (TotemToken.sol#170)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        State variables written after the call(s):
        - _transfer(sender,recipient,tokensToTransfer) (TotemToken.sol#170)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (@openzeppelin\contracts\tok
en\ERC20\ERC20.sol#214)
                - _balances[recipient] = _balances[recipient].add(amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#215)
Reentrancy in StakingPool.claimReward() (Staking\StakingPool.sol#170-194):
        External calls:
        - rewardManager.rewardUser(_msgSender(),reward) (Staking\StakingPool.sol#175)
        - require(bool)(btcToken.transfer(_msgSender(),btcReward)) (Staking\StakingPool.sol#178)
        State variables written after the call(s):
        - _withdrawStakingReward(_msgSender()) (Staking\StakingPool.sol#180)
                - userStakes[i].lastWithdrawalTime = block.timestamp (Staking\StakingPool.sol#399)
```

```
                - userStakes[i].lastWithdrawalTime = block.timestamp (Staking\StakingPool.sol#399)
                - userStakes[i].amountWithdrawn = userStakes[i].amountWithdrawn.add(rewardPerStake) (Staking\StakingPool.sol#400-402)
        - _withdrawPrizeReward(_msgSender()) (Staking\StakingPool.sol#181)
                - userStakes[i].prizeRewardWithdrawn = true (Staking\StakingPool.sol#476)
Reentrancy in StakingPool.claimReward() (Staking\StakingPool.sol#170-194):
        External calls:
        - rewardManager.rewardUser(_msgSender(),reward) (Staking\StakingPool.sol#175)
        - require(bool)(btcToken.transfer(_msgSender(),btcReward)) (Staking\StakingPool.sol#178)
        - totemToken.transfer(_msgSender(),stakedBalance) (Staking\StakingPool.sol#186)
        State variables written after the call(s):
        - _withdrawStakedBalance(_msgSender()) (Staking\StakingPool.sol#187)
                - userStakes[i].didUnstake = true (Staking\StakingPool.sol#505)
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
        - _transfer(address(this),CommunityDevelopmentAddr,COMMUNITY_DEVELOPMENT) (TotemToken.sol#103-107)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        State variables written after the call(s):
        - _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (@openzeppelin\contracts\tok
en\ERC20\ERC20.sol#214)
                - _balances[recipient] = _balances[recipient].add(amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#215)
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
        - _transfer(address(this),CommunityDevelopmentAddr,COMMUNITY_DEVELOPMENT) (TotemToken.sol#103-107)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
```

```
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),LiquidityPoolAddr,LIQUIDITY_POOL) (TotemToken.sol#109)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        State variables written after the call(s):
        - _transfer(address(this),LiquidityPoolAddr,LIQUIDITY_POOL) (TotemToken.sol#109)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (@openzeppelin\contracts\tok
en\ERC20\ERC20.sol#214)
                - _balances[recipient] = _balances[recipient].add(amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#215)
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
        - _transfer(address(this),CommunityDevelopmentAddr,COMMUNITY_DEVELOPMENT) (TotemToken.sol#103-107)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),LiquidityPoolAddr,LIQUIDITY_POOL) (TotemToken.sol#109)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PublicSaleAddr,PUBLIC_SALE) (TotemToken.sol#110)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        State variables written after the call(s):
        - _transfer(address(this),PublicSaleAddr,PUBLIC_SALE) (TotemToken.sol#110)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (@openzeppelin\contracts\tok
en\ERC20\ERC20.sol#214)
                - _balances[recipient] = _balances[recipient].add(amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#215)
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
```

```
External calls:
- _transfer(address(this),CommunityDevelopmentAddr,COMMUNITY_DEVELOPMENT) (TotemToken.sol#103-107)
        - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
- _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
        - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
- _transfer(address(this),LiquidityPoolAddr,LIQUIDITY_POOL) (TotemToken.sol#109)
        - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
- _transfer(address(this),PublicSaleAddr,PUBLIC_SALE) (TotemToken.sol#110)
        - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
- _transfer(address(this),AdvisorsAddr,ADVISORS) (TotemToken.sol#111)
        - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
State variables written after the call(s):
-  transfer(address(this),AdvisorsAddr,ADVISORS) (TotemToken.sol#111)
        - _transfer(address(this),AdvisorsAddr,ADVISORS) (TotemToken.sol#111)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (@openzeppelin\contracts\tok
en\ERC20\ERC20.sol#214)
                - _balances[recipient] = _balances[recipient].add(amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#215)
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
        - _transfer(address(this),CommunityDevelopmentAddr,COMMUNITY_DEVELOPMENT) (TotemToken.sol#103-107)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),LiquidityPoolAddr,LIQUIDITY_POOL) (TotemToken.sol#109)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PublicSaleAddr,PUBLIC_SALE) (TotemToken.sol#110)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),AdvisorsAddr,ADVISORS) (TotemToken.sol#111)
```

```
        - _transfer(address(this),AdvisorsAddr,ADVISORS) (TotemToken.sol#111)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),SeedInvestmentAddr,SEED_INVESTMENT) (TotemToken.sol#112)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        State variables written after the call(s):
        - _transfer(address(this),SeedInvestmentAddr,SEED_INVESTMENT) (TotemToken.sol#112)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (@openzeppelin\contracts\tok
en\ERC20\ERC20.sol#214)
                - _balances[recipient] = _balances[recipient].add(amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#215)
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
        - _transfer(address(this),CommunityDevelopmentAddr,COMMUNITY_DEVELOPMENT) (TotemToken.sol#103-107)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),LiquidityPoolAddr,LIQUIDITY_POOL) (TotemToken.sol#109)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PublicSaleAddr,PUBLIC_SALE) (TotemToken.sol#110)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),AdvisorsAddr,ADVISORS) (TotemToken.sol#111)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),SeedInvestmentAddr,SEED_INVESTMENT) (TotemToken.sol#112)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PrivateSaleAddr,PRIVATE_SALE) (TotemToken.sol#113)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        State variables written after the call(s):
        - _transfer(address(this),PrivateSaleAddr,PRIVATE_SALE) (TotemToken.sol#113)
```

```
        State variables written after the call(s):
        - _transfer(address(this),PrivateSaleAddr,PRIVATE_SALE) (TotemToken.sol#113)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (@openzeppelin\contracts\tok
en\ERC20\ERC20.sol#214)
                - _balances[recipient] = _balances[recipient].add(amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#215)
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
        - _transfer(address(this),CommunityDevelopmentAddr,COMMUNITY_DEVELOPMENT) (TotemToken.sol#103-107)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),LiquidityPoolAddr,LIQUIDITY_POOL) (TotemToken.sol#109)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PublicSaleAddr,PUBLIC_SALE) (TotemToken.sol#110)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),AdvisorsAddr,ADVISORS) (TotemToken.sol#111)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),SeedInvestmentAddr,SEED_INVESTMENT) (TotemToken.sol#112)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PrivateSaleAddr,PRIVATE_SALE) (TotemToken.sol#113)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),TeamAllocationAddr,TEAM_ALLOCATION) (TotemToken.sol#114)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        State variables written after the call(s):
        - _transfer(address(this),TeamAllocationAddr,TEAM_ALLOCATION) (TotemToken.sol#114)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (@openzeppelin\contracts\tok
en\ERC20\ERC20.sol#214)
```

```
en\ERC20\ERC20.sol#214)
                - _balances[recipient] = _balances[recipient].add(amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#215)
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
        - _transfer(address(this),CommunityDevelopmentAddr,COMMUNITY_DEVELOPMENT) (TotemToken.sol#103-107)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),LiquidityPoolAddr,LIQUIDITY_POOL) (TotemToken.sol#109)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PublicSaleAddr,PUBLIC_SALE) (TotemToken.sol#110)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),AdvisorsAddr,ADVISORS) (TotemToken.sol#111)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),SeedInvestmentAddr,SEED_INVESTMENT) (TotemToken.sol#112)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PrivateSaleAddr,PRIVATE_SALE) (TotemToken.sol#113)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),TeamAllocationAddr,TEAM_ALLOCATION) (TotemToken.sol#114)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StrategicRoundAddr,STRATEGIC_ROUND) (TotemToken.sol#115)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        State variables written after the call(s):
        - _transfer(address(this),StrategicRoundAddr,STRATEGIC_ROUND) (TotemToken.sol#115)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (@openzeppelin\contracts\tok
en\ERC20\ERC20.sol#214)
                - _balances[recipient] = _balances[recipient].add(amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#215)
```

```
en\ERC20\ERC20.sol#214)
                - _balances[recipient] = _balances[recipient].add(amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#215)
        - _isDistributionComplete = true (TotemToken.sol#128)
Reentrancy in StakingPool.stake(uint256,uint256) (Staking\StakingPool.sol#120-168):
        External calls:
        - totemToken.transferFrom(_msgSender(),address(this),_amount) (Staking\StakingPool.sol#136)
        - totemToken.transfer(totemToken.taxationWallet(),stakeTaxAmount) (Staking\StakingPool.sol#140)
        State variables written after the call(s):
        - _launchPool() (Staking\StakingPool.sol#164)
                - isLocked = true (Staking\StakingPool.sol#277)
        - totalStaked = totalStaked.add(_amount) (Staking\StakingPool.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
BTCDistributor.transferTokensThroughSwap(address,uint256,uint256,uint256) (Distribution\BTCDistributor.sol#28-43) ignores return value
by swapRouter.swapExactTokensForTokens(_usdAmount,_btcAmount,getPathForUSDToBTC(),_to,_deadline) (Distribution\BTCDistributor.sol#36-42
)
TotemToken._transfer(address,address,uint256) (TotemToken.sol#146-155) ignores return value by locker.lockOrGetPenalty(sender,recipient
) (TotemToken.sol#152)
USDRetriever.approveTokens(address,uint256) (Distribution\USDRetriever.sol#17-20) ignores return value by USDCContract.approve(_to,_amo
unt) (Distribution\USDRetriever.sol#18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
BTCDistributor.constructor(address,address,address).USDCContractAddress (Distribution\BTCDistributor.sol#15) lacks a zero-check on :
                - USDC_CONTRACT_ADDRESS = USDCContractAddress (Distribution\BTCDistributor.sol#19)
BTCDistributor.constructor(address,address,address).wBTCContractAddress (Distribution\BTCDistributor.sol#16) lacks a zero-check on :
                - WBTC_CONTRACT_ADDRESS = wBTCContractAddress (Distribution\BTCDistributor.sol#20)
StakingPool.constructor(TotemToken,RewardManager,address,address,address,address,address,uint256[7])._poolCreator (Staking\StakingPool.
```

```
StakingPool.constructor(TotemToken,RewardManager,address,address,address,address,address,uint256[7])._poolCreator (Staking\StakingPool.
sol#84) lacks a zero-check on :
                - poolCreator = _poolCreator (Staking\StakingPool.sol#97)
StakingPoolFactory.constructor(TotemToken,RewardManager,address,address,address,address)._oracleContract (Staking\StakingPoolFactory.so
l#32) lacks a zero-check on :
                - oracleContract = _oracleContract (Staking\StakingPoolFactory.sol#39)
StakingPoolFactory.constructor(TotemToken,RewardManager,address,address,address,address)._swapRouter (Staking\StakingPoolFactory.sol#33
) lacks a zero-check on :
                - swapRouter = _swapRouter (Staking\StakingPoolFactory.sol#40)
StakingPoolFactory.constructor(TotemToken,RewardManager,address,address,address,address)._usdToken (Staking\StakingPoolFactory.sol#34)
lacks a zero-check on :
                - usdToken = _usdToken (Staking\StakingPoolFactory.sol#41)
StakingPoolFactory.constructor(TotemToken,RewardManager,address,address,address,address)._btcToken (Staking\StakingPoolFactory.sol#35)
lacks a zero-check on :
                - btcToken = _btcToken (Staking\StakingPoolFactory.sol#42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in TotemToken._transfer(address,address,uint256) (TotemToken.sol#146-155):
        External calls:
        - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        State variables written after the call(s):
        - ERC20._transfer(sender,recipient,amount) (TotemToken.sol#154)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (@openzeppelin\contracts\tok
en\ERC20\ERC20.sol#214)
                - _balances[recipient] = _balances[recipient].add(amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#215)
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
```

```
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
        - _transfer(address(this),CommunityDevelopmentAddr,COMMUNITY_DEVELOPMENT) (TotemToken.sol#103-107)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),LiquidityPoolAddr,LIQUIDITY_POOL) (TotemToken.sol#109)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PublicSaleAddr,PUBLIC_SALE) (TotemToken.sol#110)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#216)
                -  transfer(address(this),PublicSaleAddr,PUBLIC_SALE) (TotemToken.sol#110)
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
        - _transfer(address(this),CommunityDevelopmentAddr,COMMUNITY_DEVELOPMENT) (TotemToken.sol#103-107)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),LiquidityPoolAddr,LIQUIDITY_POOL) (TotemToken.sol#109)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PublicSaleAddr,PUBLIC_SALE) (TotemToken.sol#110)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),AdvisorsAddr,ADVISORS) (TotemToken.sol#111)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#216)
```

```
        - Transfer(sender,recipient,amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#216)
                - _transfer(address(this),AdvisorsAddr,ADVISORS) (TotemToken.sol#111)
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
        - _transfer(address(this),CommunityDevelopmentAddr,COMMUNITY_DEVELOPMENT) (TotemToken.sol#103-107)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),LiquidityPoolAddr,LIQUIDITY_POOL) (TotemToken.sol#109)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PublicSaleAddr,PUBLIC_SALE) (TotemToken.sol#110)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),AdvisorsAddr,ADVISORS) (TotemToken.sol#111)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),SeedInvestmentAddr,SEED_INVESTMENT) (TotemToken.sol#112)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#216)
                - _transfer(address(this),SeedInvestmentAddr,SEED_INVESTMENT) (TotemToken.sol#112)
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
        - _transfer(address(this),CommunityDevelopmentAddr,COMMUNITY_DEVELOPMENT) (TotemToken.sol#103-107)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),LiquidityPoolAddr,LIQUIDITY_POOL) (TotemToken.sol#109)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
```

```
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PublicSaleAddr,PUBLIC_SALE) (TotemToken.sol#110)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),AdvisorsAddr,ADVISORS) (TotemToken.sol#111)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),SeedInvestmentAddr,SEED_INVESTMENT) (TotemToken.sol#112)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PrivateSaleAddr,PRIVATE_SALE) (TotemToken.sol#113)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#216)
                - _transfer(address(this),PrivateSaleAddr,PRIVATE_SALE) (TotemToken.sol#113)
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
        - _transfer(address(this),CommunityDevelopmentAddr,COMMUNITY_DEVELOPMENT) (TotemToken.sol#103-107)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),LiquidityPoolAddr,LIQUIDITY_POOL) (TotemToken.sol#109)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PublicSaleAddr,PUBLIC_SALE) (TotemToken.sol#110)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),AdvisorsAddr,ADVISORS) (TotemToken.sol#111)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),SeedInvestmentAddr,SEED_INVESTMENT) (TotemToken.sol#112)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PrivateSaleAddr,PRIVATE_SALE) (TotemToken.sol#113)
```

```
        - _transfer(address(this),PrivateSaleAddr,PRIVATE_SALE) (TotemToken.sol#113)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),TeamAllocationAddr,TEAM_ALLOCATION) (TotemToken.sol#114)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#216)
                - _transfer(address(this),TeamAllocationAddr,TEAM_ALLOCATION) (TotemToken.sol#114)
Reentrancy in TotemToken.distributeTokens() (TotemToken.sol#100-129):
        External calls:
        - _transfer(address(this),CommunityDevelopmentAddr,COMMUNITY_DEVELOPMENT) (TotemToken.sol#103-107)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StakingRewardsAddr,STAKING_REWARDS) (TotemToken.sol#108)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),LiquidityPoolAddr,LIQUIDITY_POOL) (TotemToken.sol#109)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PublicSaleAddr,PUBLIC_SALE) (TotemToken.sol#110)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),AdvisorsAddr,ADVISORS) (TotemToken.sol#111)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),SeedInvestmentAddr,SEED_INVESTMENT) (TotemToken.sol#112)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),PrivateSaleAddr,PRIVATE_SALE) (TotemToken.sol#113)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),TeamAllocationAddr,TEAM_ALLOCATION) (TotemToken.sol#114)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(address(this),StrategicRoundAddr,STRATEGIC_ROUND) (TotemToken.sol#115)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
```

```
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#216)
                - _transfer(address(this),StrategicRoundAddr,STRATEGIC_ROUND) (TotemToken.sol#115)
Reentrancy in StakingPool.stake(uint256,uint256) (Staking\StakingPool.sol#120-168):
        External calls:
        - totemToken.transferFrom(_msgSender(),address(this),_amount) (Staking\StakingPool.sol#136)
        - totemToken.transfer(totemToken.taxationWallet(),stakeTaxAmount) (Staking\StakingPool.sol#140)
        Event emitted after the call(s):
        - PoolLaunched() (Staking\StakingPool.sol#279)
                - _launchPool() (Staking\StakingPool.sol#164)
        - Stake(_msgSender(),_amount,_pricePrediction) (Staking\StakingPool.sol#167)
Reentrancy in TotemToken.transferFrom(address,address,uint256) (TotemToken.sol#185-202):
        External calls:
        - _transferWithTax(sender,recipient,amount) (TotemToken.sol#190-192)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        - _transfer(sender,recipient,amount) (TotemToken.sol#190-192)
                - locker.lockOrGetPenalty(sender,recipient) (TotemToken.sol#152)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (@openzeppelin\contracts\token\ERC20\ERC20.sol#277)
                - approve(_msgSender(),allowance(sender,_msgSender()).sub(amount,Transfer amount exceeds allowance)) (TotemToken.sol#19
4-200)
Reentrancy in TotemVesting.withdraw() (TotemVesting\TotemVesting.sol#145-157):
        External calls:
        - require(bool)(totemToken.transfer(_msgSender(),_withdrawable)) (TotemVesting\TotemVesting.sol#154)
        Event emitted after the call(s):
        - Withdraw(_msgSender(),_withdrawable) (TotemVesting\TotemVesting.sol#155)
```

```
        - Withdraw(_msgSender(),_withdrawable) (TotemVesting\TotemVesting.sol#155)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
PrivateSaleVesting.vested(address) (TotemVesting\PrivateSaleVesting.sol#30-82) uses timestamp for comparisons
        Dangerous comparisons:
        - ! isStartTimeSet || (_vestingSchedule.totalAmount == 0) || (lockPeriods == 0 && releasePeriods == 0) || (block.timestamp < st
artTime) (TotemVesting\PrivateSaleVesting.sol#38-41)
        - block.timestamp < startTime.add(endLock) (TotemVesting\PrivateSaleVesting.sol#47)
        - block.timestamp >= startTime.add(_end) (TotemVesting\PrivateSaleVesting.sol#52)
        - period <= lockPeriods (TotemVesting\PrivateSaleVesting.sol#58)
        - period >= lockPeriods.add(releasePeriods) (TotemVesting\PrivateSaleVesting.sol#61)
        - period.sub(lockPeriods) == 1 (TotemVesting\PrivateSaleVesting.sol#68)
StakingPool.claimReward() (Staking\StakingPool.sol#170-194) uses timestamp for comparisons
        Dangerous comparisons:
        - reward > 0 (Staking\StakingPool.sol#173)
        - totemToken.balanceOf(address(rewardManager)) >= reward (Staking\StakingPool.sol#174)
        - btcReward > 0 (Staking\StakingPool.sol#178)
        - require(bool)(btcToken.transfer(_msgSender(),btcReward)) (Staking\StakingPool.sol#178)
StakingPool.purchaseBTC(uint256,uint256) (Staking\StakingPool.sol#196-220) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool)(deadline >= block.timestamp) (Staking\StakingPool.sol#202)
StakingPool._getStakingRewardPerStake(address,uint256) (Staking\StakingPool.sol#351-376) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > (startDate.add(launchTime).add(maturityTime)) (Staking\StakingPool.sol#359-362)
StakingPool._getEnhancedRewardRate(uint256) (Staking\StakingPool.sol#406-427) uses timestamp for comparisons
        Dangerous comparisons:
        - difference < 172800 (Staking\StakingPool.sol#414)
```

```
        - difference < 172800 (Staking\StakingPool.sol#414)
        - difference < 259200 (Staking\StakingPool.sol#416)
        - difference < 345600 (Staking\StakingPool.sol#418)
        - difference < 432000 (Staking\StakingPool.sol#420)
        - difference < 518400 (Staking\StakingPool.sol#422)
TotemVesting.addRecipient(address,uint256) (TotemVesting\TotemVesting.sol#58-82) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool)(! isStartTimeSet || startTime > block.timestamp) (TotemVesting\TotemVesting.sol#63)
TotemVesting.setStartTime(uint256) (TotemVesting\TotemVesting.sol#84-93) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool)(! isStartTimeSet || startTime > block.timestamp) (TotemVesting\TotemVesting.sol#86)
        - require(bool)(_newStartTime > block.timestamp) (TotemVesting\TotemVesting.sol#87)
TotemVesting.vested(address) (TotemVesting\TotemVesting.sol#96-135) uses timestamp for comparisons
        Dangerous comparisons:
        - ! isStartTimeSet || (_vestingSchedule.totalAmount == 0) || (lockPeriods == 0 && releasePeriods == 0) || (block.timestamp < st
artTime) (TotemVesting\TotemVesting.sol#104-107)
        - block.timestamp < startTime.add(endLock) (TotemVesting\TotemVesting.sol#113)
        - block.timestamp >= startTime.add(_end) (TotemVesting\TotemVesting.sol#118)
        - period <= lockPeriods (TotemVesting\TotemVesting.sol#124)
        - period >= lockPeriods.add(releasePeriods) (TotemVesting\TotemVesting.sol#127)
TotemVesting.withdraw() (TotemVesting\TotemVesting.sol#145-157) uses timestamp for comparisons
        Dangerous comparisons:
        - _withdrawable > 0 (TotemVesting\TotemVesting.sol#153)
        - require(bool)(totemToken.transfer(_msgSender(),_withdrawable)) (TotemVesting\TotemVesting.sol#154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Different versions of Solidity is used:
```

```
Different versions of Solidity is used:
        - Version used: ['0.7.6', '>=0.4.22<0.9.0', '>=0.6.0', '>=0.6.0<0.8.0']
        - 0.7.6 (TotemVesting\AdvisorsVesting.sol#2)
        - >=0.6.0 (@chainlink\contracts\src\v0.6\interfaces\AggregatorV3Interface.sol#2)
        - 0.7.6 (BasisPoints.sol#2)
        - 0.7.6 (Distribution\BTCDistributor.sol#2)
        - 0.7.6 (TotemVesting\CommunityVesting.sol#2)
        - >=0.6.0<0.8.0 (@openzeppelin\contracts\utils\Context.sol#3)
        - >=0.6.0<0.8.0 (@openzeppelin\contracts\token\ERC20\ERC20.sol#3)
        - >=0.6.0<0.8.0 (@openzeppelin\contracts\token\ERC20\IERC20.sol#3)
        - 0.7.6 (ILocker.sol#2)
        - 0.7.6 (PancakeSwap\IPancakeRouter.sol#2)
        - >=0.4.22<0.9.0 (Migrations.sol#2)
        - 0.7.6 (Role\Operator.sol#2)
        - >=0.6.0<0.8.0 (@openzeppelin\contracts\access\Ownable.sol#3)
        - 0.7.6 (Role\PoolCreator.sol#2)
        - 0.7.6 (Price\PriceConsumer.sol#2)
        - 0.7.6 (TotemVesting\PrivateSaleVesting.sol#2)
        - 0.7.6 (Role\Rewarder.sol#2)
        - 0.7.6 (Staking\RewardManager.sol#2)
        - 0.7.6 (Role\Roles.sol#2)
        - >=0.6.0<0.8.0 (@openzeppelin\contracts\math\SafeMath.sol#3)
        - 0.7.6 (TotemVesting\SeedVesting.sol#2)
        - 0.7.6 (Staking\StakingPool.sol#2)
        - 0.7.6 (Staking\StakingPoolFactory.sol#2)
        - 0.7.6 (TotemVesting\StrategicVesting.sol#2)
        - 0.7.6 (TotemVesting\TeamVesting.sol#2)
```

```
        - 0.7.6 (Test\WrappedERC20Token.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
BasisPoints.addBP(uint256,uint256) (BasisPoints.sol#20-24) is never used and should be removed
BasisPoints.divBP(uint256,uint256) (BasisPoints.sol#15-18) is never used and should be removed
BasisPoints.subBP(uint256,uint256) (BasisPoints.sol#26-30) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0 (@chainlink\contracts\src\v0.6\interfaces\AggregatorV3Interface.sol#2) allows old versions
Pragma version>=0.6.0<0.8.0 (@openzeppelin\contracts\utils\Context.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (@openzeppelin\contracts\token\ERC20\ERC20.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (@openzeppelin\contracts\token\ERC20\IERC20.sol#3) is too complex
Pragma version>=0.4.22<0.9.0 (Migrations.sol#2) is too complex
Pragma version>=0.6.0<0.8.0 (@openzeppelin\contracts\access\Ownable.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (@openzeppelin\contracts\math\SafeMath.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter BTCDistributor.transferTokensThroughSwap(address,uint256,uint256,uint256)._to (Distribution\BTCDistributor.sol#29) is not in
mixedCase
Parameter BTCDistributor.transferTokensThroughSwap(address,uint256,uint256,uint256)._usdAmount (Distribution\BTCDistributor.sol#30) is
not in mixedCase
Parameter BTCDistributor.transferTokensThroughSwap(address,uint256,uint256,uint256)._btcAmount (Distribution\BTCDistributor.sol#31) is
not in mixedCase
Parameter BTCDistributor.transferTokensThroughSwap(address,uint256,uint256,uint256)._deadline (Distribution\BTCDistributor.sol#32) is n
ot in mixedCase
Parameter BTCDistributor.getEstimatedBTCForUSD(uint256)._amount (Distribution\BTCDistributor.sol#48) is not in mixedCase
Variable BTCDistributor.USDC_CONTRACT_ADDRESS (Distribution\BTCDistributor.sol#8) is not in mixedCase
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
RewardManager.operator (Staking\RewardManager.sol#11) is never used in RewardManager (Staking\RewardManager.sol#9-45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
RewardManager.operator (Staking\RewardManager.sol#11) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (@openzeppelin\contracts\token\ERC20\ERC20.sol#64-66)
symbol() should be declared external:
        - ERC20.symbol() (@openzeppelin\contracts\token\ERC20\ERC20.sol#72-74)
decimals() should be declared external:
        - ERC20.decimals() (@openzeppelin\contracts\token\ERC20\ERC20.sol#89-91)
totalSupply() should be declared external:
        - ERC20.totalSupply() (@openzeppelin\contracts\token\ERC20\ERC20.sol#96-98)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (@openzeppelin\contracts\token\ERC20\ERC20.sol#103-105)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#115-118)
        - TotemToken.transfer(address,uint256) (TotemToken.sol#173-183)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#152-156)
        - TotemToken.transferFrom(address,address,uint256) (TotemToken.sol#185-202)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#170-173)
decreaseAllowance(address,uint256) should be declared external:
```

```
        - ERC20.transfer(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#115-118)
        - TotemToken.transfer(address,uint256) (TotemToken.sol#173-183)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#152-156)
        - TotemToken.transferFrom(address,address,uint256) (TotemToken.sol#185-202)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#170-173)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#189-192)
setCompleted(uint256) should be declared external:
        - Migrations.setCompleted(uint256) (Migrations.sol#16-18)
renounceOperator() should be declared external:
        - Operator.renounceOperator() (Role\Operator.sol#38-40)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (@openzeppelin\contracts\access\Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (@openzeppelin\contracts\access\Ownable.sol#63-67)
addPoolCreator(address) should be declared external:
        - PoolCreator.addPoolCreator(address) (Role\PoolCreator.sol#33-35)
renouncePoolCreator() should be declared external:
        - PoolCreator.renouncePoolCreator() (Role\PoolCreator.sol#37-39)
renounceRewarder() should be declared external:
        - Rewarder.renounceRewarder() (Role\Rewarder.sol#38-40)
setOperator(address) should be declared external:
        - RewardManager.setOperator(address) (Staking\RewardManager.sol#20-28)
addPool(address) should be declared external:
        - RewardManager.addPool(address) (Staking\RewardManager.sol#30-38)
```

```
addPool(address) should be declared external:
        - RewardManager.addPool(address) (Staking\RewardManager.sol#30-38)
rewardUser(address,uint256) should be declared external:
        - RewardManager.rewardUser(address,uint256) (Staking\RewardManager.sol#40-44)
launchPool() should be declared external:
        - StakingPool.launchPool() (Staking\StakingPool.sol#272-274)
setDistributionTeamsAddresses(address,address,address,address,address,address,address,address,address) should be declared external:
        - TotemToken.setDistributionTeamsAddresses(address,address,address,address,address,address,address,address,address) (TotemToken
.sol#66-98)
distributeTokens() should be declared external:
        - TotemToken.distributeTokens() (TotemToken.sol#100-129)
setTaxRate(uint256) should be declared external:
        - TotemToken.setTaxRate(uint256) (TotemToken.sol#131-134)
setTaxationWallet(address) should be declared external:
        - TotemToken.setTaxationWallet(address) (TotemToken.sol#141-144)
addRecipient(address,uint256) should be declared external:
        - TotemVesting.addRecipient(address,uint256) (TotemVesting\TotemVesting.sol#58-82)
setStartTime(uint256) should be declared external:
        - TotemVesting.setStartTime(uint256) (TotemVesting\TotemVesting.sol#84-93)
withdraw() should be declared external:
        - TotemVesting.withdraw() (TotemVesting\TotemVesting.sol#145-157)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:. analyzed (31 contracts with 75 detectors), 188 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Disclaimer

QuillHash audit is not a security warranty, investment advice, or an endorsement of the Totem contracts. Securing smart contracts is a multistep process; therefore, running a bug bounty program as a complement to this audit is strongly recommended.

Closing Summary

The use case of the smart contract is very well designed and implemented. Altogether, the code is well written and demonstrates effective use of abstraction, separation of concerns, and modularity. The majority of the concerns addressed above have been acknowledged, implemented and verified.

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com