OBELISK

Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 52	2021-11-25	DoD4uFN, Mechwar	Audit Final

Audit Notes

Audit Date	2021-10-03 - 2021-11-25
Auditor/Auditors	DoD4uFN, Mechwar
Auditor/Auditors Contact Information	contact@obeliskauditing.com
Notes	Specified code and contracts are audited for security flaws. UI/UX (website), logic, team, and tokenomics are not audited.
Audit Report Number	OB585858511

Disclaimer

This audit is not financial, investment, or any other kind of advice and is for informational purposes only. This report is not a substitute for doing your own research and due diligence. Obelisk is not responsible or liable for any loss, damage, or otherwise caused by reliance on this report for any purpose. Obelisk has based this audit report solely on the information provided by the audited party and on facts that existed before or during the audit being conducted. Obelisk is not responsible for any outcome, including changes done to the contract/contracts after the audit was published. This audit is fully objective and only discerns what the contract is saying without adding any opinion to it. The audit is paid by the project but neither the auditors nor Obelisk has any other connection to the project and has no obligations other than to publish an objective report. Obelisk will always publish its findings regardless of the outcome of the findings. The audit only covers the subject areas detailed in this report and unless specifically stated, nothing else has been audited. Obelisk assumes that the provided information and material were not altered, suppressed, or misleading. This report is published by Obelisk, and Obelisk has sole ownership of this report. Use of this report for any reason other than for informational purposes on the subjects reviewed in this report including the use of any part of this report is prohibited without the express written consent of Obelisk.

Obelisk Auditing

Defi is a relatively new concept but has seen exponential growth to a point where there is a multitude of new projects created every day. In a fast-paced world like this, there will also be an enormous amount of scams. The scams have become so elaborate that it's hard for the common investor to trust a project, even though it could be legit. We saw a need for creating high-quality audits at a fast phase to keep up with the constantly expanding market. With the Obelisk stamp of approval, a legitimate project can easily grow its user base exponentially in a world where trust means everything. Obelisk Auditing consists of a group of security experts that specialize in security and structural operations, with previous work experience from among other things, PricewaterhouseCoopers. All our audits will always be conducted by at least two independent auditors for maximum security and professionalism.

As a comprehensive security firm, Obelisk provides all kinds of audits and project assistance.

Audit Information

The auditors always conducted a manual visual inspection of the code to find security flaws that automatic tests would not find. Comprehensive tests are also conducted in a specific test environment that utilizes exact copies of the published contract.

While conducting the audit, the Obelisk security team uses best practices to ensure that the reviewed contracts are thoroughly examined against all angles of attack. This is done by evaluating the codebase and whether it gives rise to significant risks. During the audit, Obelisk assesses the risks and assigns a risk level to each section together with an explanatory comment. Take note that the comments from the project team are their opinion and not the opinion of Obelisk.

Table of Contents

Project Information

Name	T-Node
Description	“Staking Rewards Made Easy. The era of Proof of Stake is here. Trusted Node gives you instant access to the world of staking rewards.”
Website	https://trustednode.io/
Contact	Robin#9422 on Discord
Contact information	info@trustednode.io
Token Name(s)	Trusted Node
Token Short	TNODE
Contract(s)	See Appendix A
Code Language	Solidity
Chain	Polygon / BSC

Audit of T-Node

> **The audit was conducted on not-yet-published contracts which meant that the project team could easily implement fixes to all issues found. On-chain analyses have been conducted to make sure the published contracts are the same as the audited ones.**

Obelisk was commissioned by T-Node on the 1st of October 2021 to conduct a comprehensive audit of T-Nodes' contracts. The following audit was conducted between the 3d of October 2021 and the 25th of November 2021. Two of Obelisk's security experts went through the related contracts manually using industry standards to find if any vulnerabilities could be exploited either by the project team or users.

During the audit of T-Nodes' contracts, we found multiple vulnerabilities of different risk levels. All of the vulnerabilities can be seen in this audit report. However, the project team solved all the vulnerabilities found before publishing the contracts on-chain. During the on-chain analysis, we found that the published contracts match the audited contracts including implemented fixes. However, there is no timelock on 2 important contracts, which need to be kept an eye on. Also, 2 of the contracts are unverified on-chain.

The informational findings are good to know while interacting with the project but don't directly damage the project in its current state, hence it's up to the project team if they deem that it's worth solving these issues.

The team has not reviewed the UI/UX, logic, team, or tokenomics of the T-Node project.

Please read the full document for a complete understanding of the audit.

Summary Table

Finding	ID	Severity	Status
No Limit For Protocol Values	#0001	High Risk	Closed
Staking Contract May Not Work With Same Staking And Reward Tokens	#0002	Medium Risk	Closed
Claiming Rewards Does Not Correctly Check For Available Balance	#0003	Low Risk	Closed
Transfer Fees For Staking Tokens Are Not Accounted For	#0004	Low Risk	Closed
Locking Timestamp Can Be Set By Any Address	#0005	Low Risk	Closed
Use Safe Transfer	#0006	Low Risk	Closed
Initialization Function Can Be Called Multiple Times	#0007	Low Risk	Closed
Protocol Values Should Have A View Function	#0008	Informational	Closed
No Checks To Privileged Withdraw Function	#0009	Informational	Closed
Unbound Loop	#0010	Informational	Closed
Missing Zero Checks	#0011	Informational	Closed
Multiple Contracts In One File	#0012	Informational	Closed
Compile Issue With Invalid Number Of Input Parameters	#0013	Informational	Closed
Staked Event Amount Will Always Be Incorrect	#0014	Low Risk	Closed
Changes To StakingRewardsFactory Not Updated To StakingRewards	#0015	Informational	Closed
Some Protocol Values Not Updated When Deploying StakingRewards	#0016	Informational	Closed

Compile Errors	#0017	Informational	Closed
Redundant Assignment	#0018	Informational	Closed
Loss Of Protocol Functionality	#0019	Low Risk	Closed
Contract Function Is Not Called Directly	#0020	Informational	Closed
Initialize Function Of Protocol Variables Is Not Called	#0021	Low Risk	Closed
Incorrect Check For Acceptable Reward Rate	#0022	Medium Risk	Closed
Incorrect Accounting Of The Reward Token Total Supply	#0023	Low Risk	Closed
Redundant Protocol Variable	#0024	Informational	Closed
Redundant Subtraction Of Total Rewards	#0025	Informational	Closed
Protocol Variables Should Be Public	#0026	Informational	Closed
No Timelock	#0027	Low Risk	Open
Unverified Staking Pool Contracts	#0028	Informational	Open

Findings

Manual Analysis

No Limit For Protocol Values

FINDING ID	#0001
SEVERITY	High Risk
STATUS	Closed
LOCATION	StakingRewardsFactory.sol -> 131

```
1     _lockingTimeStamp[useraddress] = lockingPeriod; //
  setting user locking ts
```

LOCATION	StakingRewardsFactory.sol -> 177

```
1   function claimRewardAmount(uint256 reward, uint256
  rewardsDuration)
```

DESCRIPTION	The *lockingPeriod* can be any period including the maximum value of *UINT256*. When the *lockingPeriod* is very large, it is impossible to recover funds from the staking token. Thus funds could be lost. The *rewardsDuration* can be arbitrarily high, causing the *rewardRate* to be effectively zero. Since the *periodFinish* cannot be reduced, a mistakenly set duration cannot be fixed.

RECOMMENDATION	Add an upper bound to the noted variables.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit e46edb43cae6a92715aca671adb3431ba8b435c7@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Staking Contract May Not Work With Same Staking And Reward Tokens

FINDING ID	#0002
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	StakingRewardsFactory.sol -> 56

```
1     rewardRate =
  rewardsToken.balanceOf(address(this)).div(rewardsDuration);
```

LOCATION	StakingRewardsFactory.sol -> 200

```
1     uint256 balance =
  rewardsToken.balanceOf(address(this));
```

DESCRIPTION	When *rewardsToken* and *stakingToken* are the same, the staked tokens can be distributed as rewards to other users if *claimRewardAmount()* or *initializeDefault()* are called.
RECOMMENDATION	Add a check for *stakingToken* not equal to *rewardsToken*.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit f56557d2109cea240ed5492ff056024dcadd82ce@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Claiming Rewards Does Not Correctly Check For Available Balance

FINDING ID	#0003
SEVERITY	Low Risk
STATUS	Closed
LOCATION	StakingRewardsFactory.sol -> 200-204

```
1     uint256 balance =
  rewardsToken.balanceOf(address(this));
2     require(
3       rewardRate <= balance.div(rewardsDuration),
4       "Provided reward too high"
5     );
```

DESCRIPTION	*claimRewardAmount* does not account for tokens already assigned for distribution when checking that enough tokens are available.
RECOMMENDATION	Ensure that tokens already assigned for distribution to staked users are not included in the balance check.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 20a828224069159d3d6c69a1d90013d5add3d8d7@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Transfer Fees For Staking Tokens Are Not Accounted For

FINDING ID	#0004
SEVERITY	Low Risk
STATUS	Closed
LOCATION	StakingRewardsFactory.sol -> 103-116 StakingRewardsFactory.sol -> 118-134

```
function stake(uint256 amount)
  external
  override
  nonReentrant
  updateReward(msg.sender)
{
  require(amount > 0, "Cannot stake 0");
  require(_lockingTimeStamp[msg.sender] <= 0);
  _totalSupply = _totalSupply.add(amount);
  _balances[msg.sender] =
_balances[msg.sender].add(amount);
  _lockingTimeStamp[msg.sender] = 0;
  stakingToken.safeTransferFrom(msg.sender,
address(this), amount);
  emit Staked(msg.sender, amount);
}
```

```
function stakeTransferWithBalance(
  uint256 amount,
  address useraddress,
  uint256 lockingPeriod
)
  external
  nonReentrant
  updateReward(useraddress)
{
  require(amount > 0, "Cannot stake 0");
  require(_balances[useraddress] <= 0, "Already staked by
user");
  _totalSupply = _totalSupply.add(amount);
  _balances[useraddress] =
_balances[useraddress].add(amount);
  _lockingTimeStamp[useraddress] = lockingPeriod; //
setting user locking ts
  stakingToken.safeTransferFrom(msg.sender,
address(this), amount);
  emit Staked(useraddress, amount);
}
```

DESCRIPTION	If the staking tokens have transfer fees, the *stake* and *stakeTransferWithBalance* functions will incorrectly update *_totalSupply* and *_balances*.
RECOMMENDATION	Check *.balanceOf* before and after the transferring of the staking token, to take into account any fees applicable.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit f56557d2109cea240ed5492ff056024dcadd82ce@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Locking Timestamp Can Be Set By Any Address

FINDING ID	#0005
SEVERITY	Low Risk
STATUS	Closed
LOCATION	StakingRewardsFactory.sol -> 118-134

```
function stakeTransferWithBalance(
  uint256 amount,
  address useraddress,
  uint256 lockingPeriod
)
  external
  nonReentrant
  updateReward(useraddress)
{
  require(amount > 0, "Cannot stake 0");
  require(_balances[useraddress] <= 0, "Already staked by user");
  _totalSupply = _totalSupply.add(amount);
  _balances[useraddress] = _balances[useraddress].add(amount);
  _lockingTimeStamp[useraddress] = lockingPeriod; // setting user locking ts
  stakingToken.safeTransferFrom(msg.sender, address(this), amount);
  emit Staked(useraddress, amount);
}
```

DESCRIPTION	*stakeTransferWithBalance* can be called by any address, and set the *_lockingTimeStamp* of any other address with a balance of 0. This will prevent staking and withdrawing from the contract.
RECOMMENDATION	Add restrictions to how *_lockingTimeStamp* is set.
RESOLUTION	The *_lockingTimeStamp* can now only be set by the *msg.sender*. Reviewed in commit

	e3e4dd9ada190ce54ec8a1bf2183244b2be041f5@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Use Safe Transfer

FINDING ID	#0006
SEVERITY	Low Risk
STATUS	Closed
LOCATION	StakingRewardsFactory.sol -> 352

```
1         IERC20(rewardsToken).transfer(info.stakingRewards,
  rewardAmount),
```

LOCATION	StakingRewardsFactory.sol -> 363

```
1     IERC20(token).transfer(msg.sender, amount);
```

DESCRIPTION	Direct transfer functions are called.
RECOMMENDATION	Use Openzeppelin's safe transfer functions. These safe transfer functions are used to catch when a transfer fails as well as unusual token behavior.
RESOLUTION	The Openzeppelin's safe transfer function was added at the appropriate locations in the contract. Reviewed in commit e3e4dd9ada190ce54ec8a1bf2183244b2be041f5@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Initialization Function Can Be Called Multiple Times

FINDING ID	#0007
SEVERITY	Low Risk
STATUS	Closed
LOCATION	StakingRewardsFactory.sol -> 52-59

```
1   function initializeDefault() external
  onlyRewardsDistribution {
2     lastUpdateTime = block.timestamp;
3     periodFinish = block.timestamp.add(rewardsDuration);
4
5     rewardRate =
  rewardsToken.balanceOf(address(this)).div(rewardsDuration);
6
7     emit DefaultInitialization();
8   }
```

DESCRIPTION	*initializeDefault* does not have a mechanism to check for subsequent calls. Multiple calls can cause *rewardRate* to distribute rewards already assigned to users.
RECOMMENDATION	Add a boolean which allows the function to be called once.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit e3e4dd9ada190ce54ec8a1bf2183244b2be041f5@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Protocol Values Should Have A View Function

FINDING ID	#0008
SEVERITY	Informational
STATUS	Closed
LOCATION	StakingRewardsFactory.sol -> 36

```
1   mapping(address => uint256) private _lockingTimeStamp;
```

DESCRIPTION	Such a variable that restricts the withdrawal action of an account should have an associated view function such that the account owner knows when it is possible to *withdraw*.
RECOMMENDATION	Create a new view function for the *_lockingTimeStamp* mapping or make it public.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit e3e4dd9ada190ce54ec8a1bf2183244b2be041f5@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

No Checks To Privileged Withdraw Function

FINDING ID	#0009
SEVERITY	Informational
STATUS	Closed
LOCATION	StakingRewardsFactory.sol -> 362-364

```
1   function pullExtraTokens(address token, uint256 amount)
  external onlyOwner {
2     IERC20(token).transfer(msg.sender, amount);
3   }
```

DESCRIPTION	*StakingRewardsFactory* contract is distributing *rewardsToken* to *StakingRewards* contracts. A bad actor can abuse the *pullExtraTokens* function to withdraw *rewardsToken*.
RECOMMENDATION	Add a check for *token* not being equal to *rewardsToken*.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit e3e4dd9ada190ce54ec8a1bf2183244b2be041f5@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Unbound Loop

FINDING ID	#0010
SEVERITY	Informational
STATUS	Closed
LOCATION	Looping over *stakingTokens.length*: • StakingRewardsFactory.sol -> 324-326

```
for (uint256 i = 0; i < stakingTokens.length; i++) {
  claimRewardAmount(stakingTokens[i]);
}
```

DESCRIPTION	Unbound loops may revert due to the gas fee limit.
RECOMMENDATION	Add an upper/lower bound parameter to the function to loop over a specific range.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 20a828224069159d3d6c69a1d90013d5add3d8d7@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Staked Event Amount Will Always Be Incorrect

FINDING ID	#00014
SEVERITY	Low Risk
STATUS	Closed
LOCATION	commit e3e4dd9ada190ce54ec8a1bf2183244b2be041f5 @staking-vault-contracts StakingRewardsFactory.sol -> 144-167

```
1   function stakeTransferWithBalance(
2     uint256 amount,
3     uint256 lockingPeriod
4   )
5     external
6     nonReentrant
7     updateReward(msg.sender)
8   {
9     require(amount > 0, "Cannot stake 0");
10    require(_balances[msg.sender] <= 0, "Already staked by
  user");
11    uint256 balance =
  stakingToken.balanceOf(address(this));
12    uint256 difference = (balance - amount);
13    amount = (balance - difference);
14    _totalSupply = _totalSupply.add(amount);
15    _balances[msg.sender] =
  _balances[msg.sender].add(amount);
16    require(lockingPeriod <= (maximumLockingPeriod +
  lockingPeriod), "Invalid locking period");
17    _lockingTimeStamp[msg.sender] = lockingPeriod; //
  setting user locking ts
18    stakingToken.safeTransferFrom(msg.sender,
  address(this), amount);
19    amount = stakingToken.balanceOf(address(this));
20    balance = stakingToken.balanceOf(address(this));
21    difference = (balance - amount);
22    amount = (balance - difference);
23    emit Staked(msg.sender, amount);
24  }
```

DESCRIPTION	The *amount* and *balance* taken from the result of

	stakingToken.balanceOf(address(this)). Thus when *balance* and *amount* are subtracted the result would always be 0. Therefore the final *amount* is incorrect and will be incorrectly passed to the *Staked* event.
RECOMMENDATION	The logic should be revised to provide the correct amount staked to the *Staked* event.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit f56557d2109cea240ed5492ff056024dcadd82ce@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Changes To StakingRewardsFactory Not Updated To StakingRewards

FINDING ID	#0015
SEVERITY	Informational
STATUS	Closed
LOCATION	commit e3e4dd9ada190ce54ec8a1bf2183244b2be041f5 @staking-vault-contracts StakingRewardsFactory.sol -> 337-356

```
function update(
  address stakingToken,
  uint256 rewardAmount,
  uint256 rewardsDuration,
  uint256 maximumLockingPeriod,
  uint256 maximumRewardsDuration
) public onlyOwner {
  StakingRewardsInfo storage info = stakingRewardsInfoByStakingToken[
    stakingToken
  ];
  require(
    info.stakingRewards != address(0),
    "StakingRewardsFactory::update: not deployed"
  );

  info.rewardAmount = rewardAmount;
  info.duration = rewardsDuration;
  info.maximumLockingPeriod = maximumLockingPeriod;
  info.maximumRewardsDuration = maximumRewardsDuration;
}
```

DESCRIPTION	Protocol values *maximumLockingPeriod* and *maximumRewardsDuration* are updated through *update* at *StakingRewardsFactory* but there is no functionality to reflect these changes at *StakingRewards*.
RECOMMENDATION	Introduce a mechanism that updates these values at *StakingRewards*.

RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 20a828224069159d3d6c69a1d90013d5add3d8d7@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Some Protocol Values Not Updated When Deploying StakingRewards

FINDING ID	#0016
SEVERITY	Informational
STATUS	Closed
LOCATION	commit e3e4dd9ada190ce54ec8a1bf2183244b2be041f5 @staking-vault-contracts StakingRewardsFactory.sol -> 314-335

```
1   function deploy(
2     address stakingToken,
3     uint256 rewardAmount,
4     uint256 rewardsDuration,
5     uint256 maximumLockingPeriod,
6     uint256 maximumRewardsDuration
7   ) public onlyOwner {
8     StakingRewardsInfo storage info =
  stakingRewardsInfoByStakingToken[
9       stakingToken
10    ];
11    require(
12      info.stakingRewards == address(0),
13      "StakingRewardsFactory::deploy: already deployed"
14    );
15
16    info.stakingRewards = address(
17      new StakingRewards(address(this), rewardsToken,
  stakingToken, rewardsDuration, maximumLockingPeriod,
  maximumRewardsDuration)
18    );
19    info.rewardAmount = rewardAmount;
20    info.duration = rewardsDuration;
21    stakingTokens.push(stakingToken);
22  }
```

DESCRIPTION	The *StakingRewardsInfo* struct members *maximumLockingPeriod* and *maximumRewardsDuration* are not updated in the *deploy* function.
RECOMMENDATION	Set the *StakingRewardsInfo* struct members

	maximumLockingPeriod and *maximumRewardsDuration* from the input parameters.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit bacd2228ddf6506d2798644c28051f1139b20d22@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Loss Of Protocol Functionality

FINDING ID	#0019
SEVERITY	Low Risk
STATUS	Closed
LOCATION	StakingRewardsFactory.sol -> 75-94

```
function update(
  address stakingToken,
  uint256 rewardAmount,
  uint256 rewardsDuration
) public onlyOwner {
  StakingRewardsInfo storage info =
stakingRewardsInfoByStakingToken[
    stakingToken
  ];
  require(
    info.stakingRewards != address(0),
    "StakingRewardsFactory::update: not deployed"
  );

  info.rewardAmount = rewardAmount;
  info.duration = rewardsDuration;
  StakingRewards(info.stakingRewards).claimRewardAmount(
      rewardAmount,
      rewardsDuration
    );
}
```

DESCRIPTION	The call to the *update* function would most likely fail due to the lack of reward token funds in the *StakingRewards* contract. This would make it impossible to update the protocol values *rewardAmount* and *duration* causing loss of protocol functionality of the *StakingRewards* and *StakingRewardsFactory* contracts.
RECOMMENDATION	The *claimRewardAmount* function in the *StakingRewards* contract should not be called here.

	A call to *StakindRewardsFactory.claimRewardAmount* would be sufficient.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 20a828224069159d3d6c69a1d90013d5add3d8d7@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Initialize Function Of Protocol Variables Is Not Called

FINDING ID	#0021
SEVERITY	Low Risk
STATUS	Closed
LOCATION	commit bacd2228ddf6506d2798644c28051f1139b20d22 @staking-vault-contracts StakingRewards.sol -> 60-72

```
function initializeDefault() external
  onlyRewardsDistribution
  nonReentrant
{
  require(isInitialized != true);
  lastUpdateTime = block.timestamp;
  periodFinish = block.timestamp.add(rewardsDuration);

  rewardRate = rewardsToken.balanceOf(address(this)).div(rewardsDuration);

  isInitialized = true;
  emit DefaultInitialization();
}
```

DESCRIPTION	*initializeDefault* function is initializing all the necessary protocol variables in order for the contract to function properly. Although, it's not being called anywhere within the contract.
RECOMMENDATION	Make sure this function is called.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit e46edb43cae6a92715aca671adb3431ba8b435c7@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE

	-token-contract

Incorrect Check For Acceptable Reward Rate

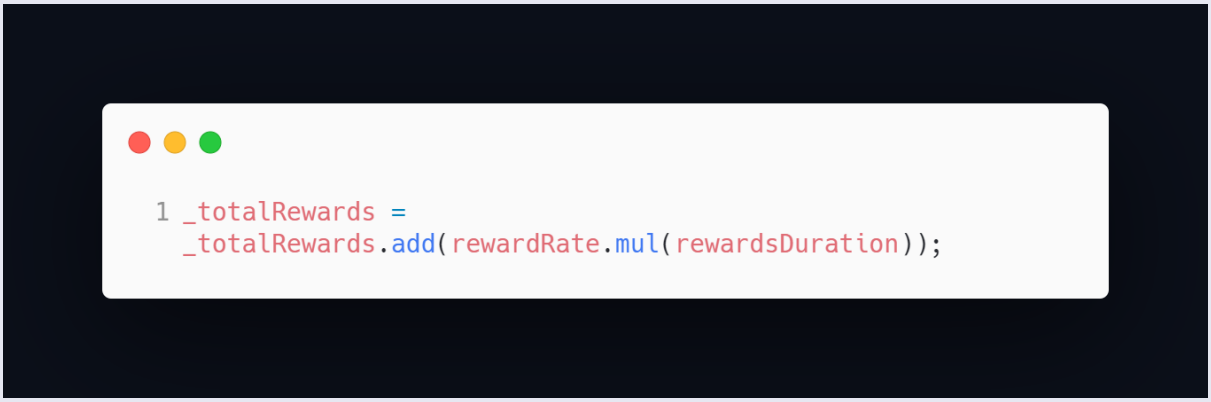
FINDING ID	#0022
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	commit e46edb43cae6a92715aca671adb3431ba8b435c7 @staking-vault-contracts StakingRewards.sol -> 241-244

```
1     require(
2       rewardRate <=
  balance.sub(rewardsAssigned).div(rewardsDuration),
3       "Provided reward too high"
4     );
```

DESCRIPTION	The prior reward duration should not be used to calculate whether the balance of reward tokens in the contract is enough to satisfy the reward rate. The prior recorded reward duration should not be used at all in the *claimRewardAmount()* function since it does not apply to the new reward amount and reward duration.
RECOMMENDATION	Instead of dividing by *rewardsDuration*, the division should use the local variable *_rewardsDuration*. Effectively changing the line of code to become: *rewardRate <= balance.sub(rewardsAssigned).div(_rewardsDuration),*
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 3d1c3a3e5659372e6eb57b60f3957b9009258ba7@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Incorrect Accounting Of The Reward Token Total Supply

FINDING ID	#0023
SEVERITY	Low Risk
STATUS	Closed
LOCATION	commit e46edb43cae6a92715aca671adb3431ba8b435c7 @staking-vault-contracts StakingRewards.sol -> 235

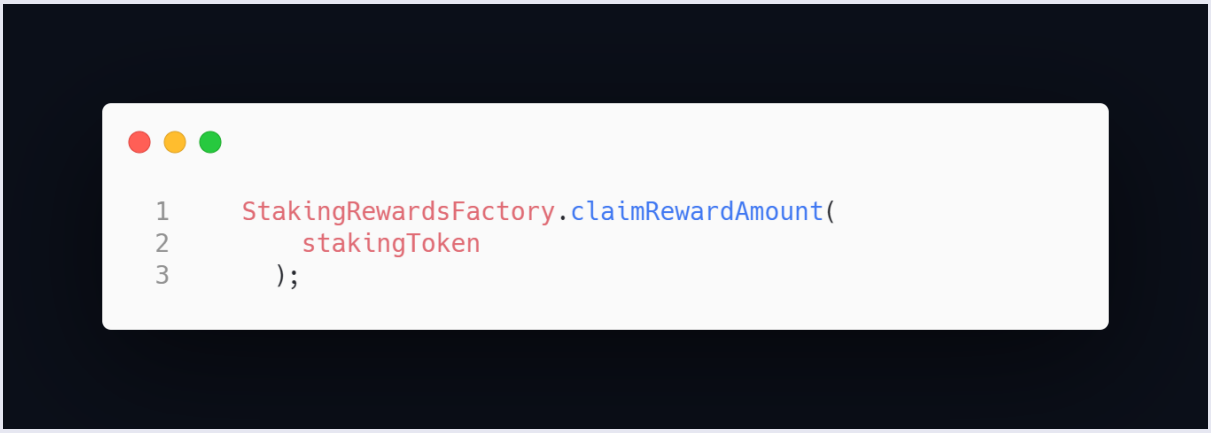


```
1 _totalRewards =
  _totalRewards.add(rewardRate.mul(rewardsDuration));
```

DESCRIPTION	The total rewards were created to track all the rewards tokens sent to the *StalkingRewards* contract. However, using the reward rate along with the duration of the reward in the case where the block timestamp is prior to the period finish would cause the total rewards to double count the prior remaining rewards. This would cause the total rewards to be larger than the actual rewards in the contract.
RECOMMENDATION	Instead of taking the reward rate and multiplying it with the duration of the reward, the *_totalRewards* should add the input parameter *reward*. Effectively changing the line of code to become: *_totalRewards = _totalRewards.add(reward);*
RESOLUTION	The project team has implemented the recommended fix by removing the *_totalRewards* protocol variable. Reviewed in commit 3d1c3a3e5659372e6eb57b60f3957b9009258ba7@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Contract Function Is Not Called Directly

FINDING ID	#0020
SEVERITY	Informational
STATUS	Closed
LOCATION	StakingRewardsFactory.sol -> 90-92



```
1    StakingRewardsFactory.claimRewardAmount(
2        stakingToken
3      );
```

DESCRIPTION	At *update* function, *claimRewardAmount* function of the same contract is being called, but it's not called directly, rather *StakingRewardsFactory.claimRewardAmount* is used.
RECOMMENDATION	Replace *StakingRewardsFactory.claimRewardAmount* with *claimRewardAmount*.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit bacd2228ddf6506d2798644c28051f1139b20d22@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Redundant Protocol Variable

FINDING ID	#0024
SEVERITY	Informational
STATUS	Closed

DESCRIPTION	*_totalRewards* actually does not participate in the contract at all and only really serves as an informational protocol value. *_totalRewards* could be removed and the contract would work fine. The reason is because of this are these lines: * StakingRewards.sol -> 69: *rewardRate = rewardsToken.balanceOf(address(this)).sub(_totalSupply).div(rewardsDuration);* * StakingRewards.sol -> 240: *uint256 balance = rewardsToken.balanceOf(address(this)).sub(_totalSupply);* This makes sure that the reward rate and its validation does not use the balances of the users' staked tokens. Also, each user's staked token is protected by the *_balances* mapping.
RECOMMENDATION	Remove *_totalRewards* protocol variable.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 3d1c3a3e5659372e6eb57b60f3957b9009258ba7@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Redundant Subtraction Of Total Rewards

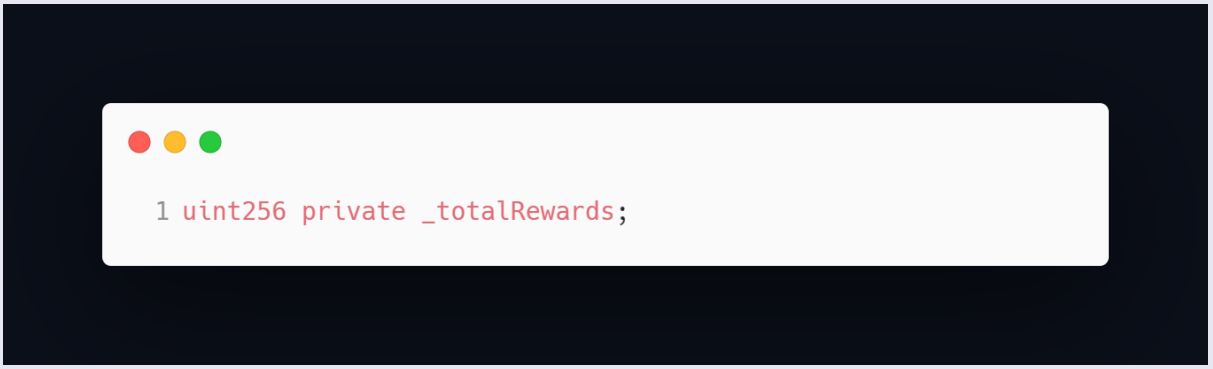
FINDING ID	#0025
SEVERITY	Informational
STATUS	Closed
LOCATION	commit e46edb43cae6a92715aca671adb3431ba8b435c7 @staking-vault-contracts StakingRewards.sol -> 131-133 & StakingRewards.sol -> 153-155

```
1     uint256 balanceBefore =
  stakingToken.balanceOf(address(this)).sub(_totalRewards);
2     stakingToken.safeTransferFrom(msg.sender,
  address(this), amount);
3     uint256 balanceAfter =
  stakingToken.balanceOf(address(this)).sub(_totalRewards);
```

DESCRIPTION	The subtracting of the total rewards from the balances is not needed since the balances are used to calculate the relative actual balance that accounts for any transfer fees. When subtracting the *balanceAfter* with the *balanceBefore* to calculate the actual *amount*, the *_totalRewards* terms cancel each other out mathematically. Therefore the subtraction of *_totalRewards* in calculating the *balanceAfter* and *balanceBefore* is redundant and can be removed.
RECOMMENDATION	Remove the subtraction of *_totalRewards* when calculating the *balanceAfter* and *balanceBefore*.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 3d1c3a3e5659372e6eb57b60f3957b9009258ba7@staking-vault-contracts

Protocol Variables Should Be Public

FINDING ID	#0026
SEVERITY	Informational
STATUS	Closed
LOCATION	commit e46edb43cae6a92715aca671adb3431ba8b435c7 @staking-vault-contracts StakingRewardsFactory.sol -> 37



```
1 uint256 private _totalRewards;
```

DESCRIPTION	The total rewards should be visible since this is the amount that would be distributed to users.
RECOMMENDATION	Create a new view function for the *_totalRewards* protocol value or make the *_totalRewards* protocol value public.
RESOLUTION	The project team has implemented the recommended fix by removing the `_totalRewards` protocol variable. Reviewed in commit 3d1c3a3e5659372e6eb57b60f3957b9009258ba7@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Static Analysis

Missing Zero Checks

FINDING ID	#0011
SEVERITY	Informational
STATUS	Closed
LOCATION	• StakingRewardsFactory.sol -> 40-50: constructor(address _rewardsDistribution, address _rewardsToken, address _stakingToken, uint256 _rewardsDuration) public • StakingRewardsFactory.sol -> 261-272: constructor(address _rewardsToken, uint256 _stakingRewardsGenesis) public Ownable()

DESCRIPTION	Functions don't check for a zero address before assigning variables.
RECOMMENDATION	Add a check for zero address if deemed necessary.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit e3e4dd9ada190ce54ec8a1bf2183244b2be041f5@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Multiple Contracts In One File

FINDING ID	#0012
SEVERITY	Informational
STATUS	Closed
LOCATION	• StakingRewardsFactory.sol -> 14: contract StakingRewards is IStakingRewards, RewardsDistributionRecipient, ReentrancyGuard • StakingRewardsFactory.sol -> 242: contract StakingRewardsFactory is Ownable

DESCRIPTION	*StakingRewardsFactory.sol* contains multiple contracts, they should be separated in their own files.
RECOMMENDATION	Have each contract in its own file, with a matching file name.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit f56557d2109cea240ed5492ff056024dcadd82ce@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Compile Issue With Invalid Number Of Input Parameters

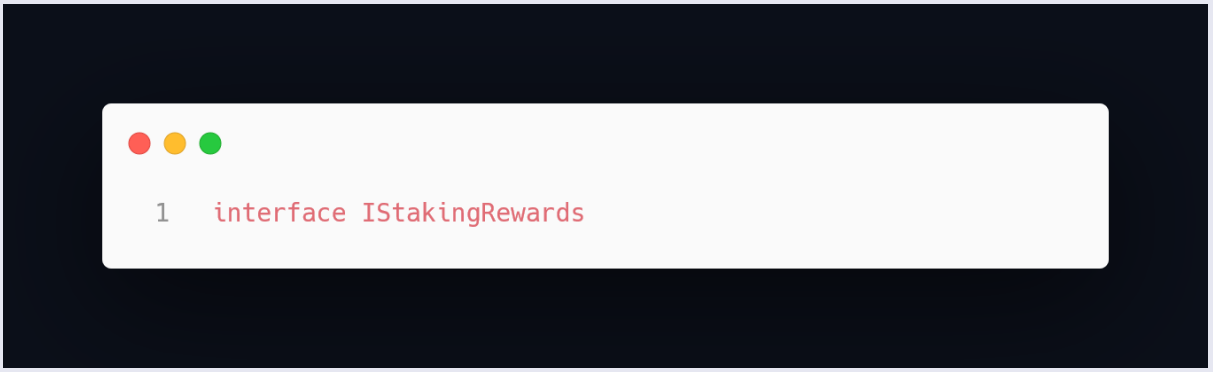
FINDING ID	#0013
SEVERITY	Informational
STATUS	Closed
LOCATION	StakingRewardsFactory.sol -> 291-293

```
1     info.stakingRewards = address(
2       new StakingRewards(address(this), rewardsToken,
  stakingToken)
3     );
```

DESCRIPTION	Can not compile due to an incorrect number of input parameters to the constructor of *StakingRewards* (expected 4, actual 3). Missing the *rewardsDuration* input parameter.
RECOMMENDATION	Add rewardsDuration to the list of arguments.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit e3e4dd9ada190ce54ec8a1bf2183244b2be041f5@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

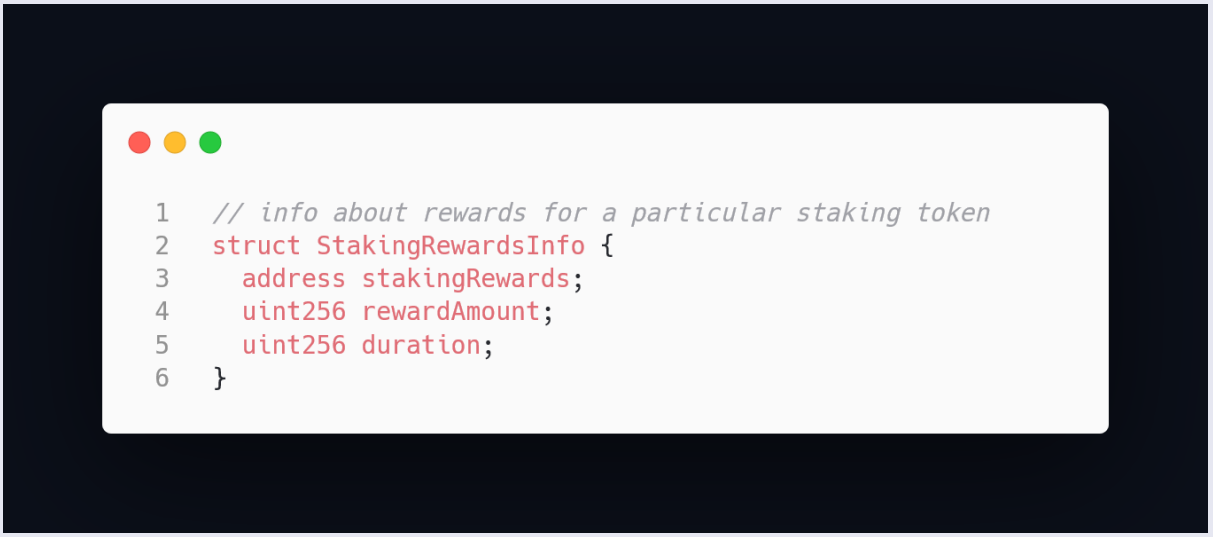
Compile Errors

FINDING ID	#0017
SEVERITY	Informational
STATUS	Closed
LOCATION	commit e3e4dd9ada190ce54ec8a1bf2183244b2be041f5 @staking-vault-contracts IStakingRewards.sol -> 4



```
1  interface IStakingRewards
```

LOCATION	commit e3e4dd9ada190ce54ec8a1bf2183244b2be041f5 @staking-vault-contracts StakingRewardsFactory.sol -> 285-290



```
1  // info about rewards for a particular staking token
2  struct StakingRewardsInfo {
3    address stakingRewards;
4    uint256 rewardAmount;
5    uint256 duration;
6  }
```

LOCATION	commit e3e4dd9ada190ce54ec8a1bf2183244b2be041f5 @staking-vault-contracts StakingRewardsFactory.sol -> 277

```
1   contract StakingRewardsFactory
```

LOCATION	commit e3e4dd9ada190ce54ec8a1bf2183244b2be041f5 @staking-vault-contracts StakingRewardsFactory.sol -> 393-396

```
1   require(
2     IERC20(rewardsToken).safeTransfer(info.stakingRewards,
  rewardAmount),
3     "StakingRewardsFactory::claimRewardAmount: transfer
  failed"
4   );
```

DESCRIPTION	Can not compile due to: • missing interface prototype view function *viewLockingTimeStamp* • missing *StakingRewardsInfo* struct members *maximumLockingPeriod* and *maximumRewardsDuration* • missing *SafeERC20* using statement for the *StakingRewardsFactory* contract (e.g. using SafeERC20 for IERC20;). • *SafeERC20.safeTransfer* not returning a value to be checked.

RECOMMENDATION	Resolve the compiler errors.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit f56557d2109cea240ed5492ff056024dcadd82ce@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

Redundant Assignment

FINDING ID	#0018
SEVERITY	Informational
STATUS	Closed
LOCATION	StakingRewardsFactory.sol -> 130-136

```
1    require(_lockingTimeStamp[msg.sender] <= 0);
2    ...
3    _lockingTimeStamp[msg.sender] = 0;
```

DESCRIPTION	The assignment to 0 here is redundant because the code won't be executed if *_lockingTimeStamp* is not equal to 0.
RECOMMENDATION	Remove this assignment.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit f56557d2109cea240ed5492ff056024dcadd82ce@staking-vault-contracts Reviewed in commit b9bb06608365d166b66293be1eb83e358a6e6952@TNODE-token-contract

On-Chain Analysis

No Timelock

FINDING ID	#0027
SEVERITY	Low Risk
STATUS	Open
LOCATION	StakingRewardsFactory.sol StakingRewardsFactory.sol (TNODE)

DESCRIPTION	The following contracts have not had their ownership transferred to a timelock contract yet: - StakingRewardsFactory.sol - StakingRewardsFactory.sol (TNODE)
RECOMMENDATION	Deploy a timelock contract and transfer the ownership to it.
RESOLUTION	N/A

Unverified Staking Pool Contracts

FINDING ID	#0028
SEVERITY	Informational
STATUS	Open
LOCATION	StakingRewards TNODE pool StakingRewards TNODE-BUSD(PCS LP) pool

DESCRIPTION	The aforementioned staking pools are unverified.
RECOMMENDATION	Verify these contracts.
RESOLUTION	N/A

Appendix A - Reviewed Documents

Document	Address
staking-vault-contracts/ RewardsDistributionRecipient.sol	0x09bF91eA8158E61116eDc0C79e6150981e81D88f 0x14b7B9e0c63a1360315b15AD5eD6Ba681eeDa836
staking-vault-contracts/ StakingRewardsFactory.sol	0x09bF91eA8158E61116eDc0C79e6150981e81D88f 0x14b7B9e0c63a1360315b15AD5eD6Ba681eeDa836
staking-vault-contracts/ IStakingRewards.sol	0x09bF91eA8158E61116eDc0C79e6150981e81D88f 0x14b7B9e0c63a1360315b15AD5eD6Ba681eeDa836
staking-vault-contracts/ StakingRewards.sol	0x09bF91eA8158E61116eDc0C79e6150981e81D88f 0x14b7B9e0c63a1360315b15AD5eD6Ba681eeDa836
StakingRewards.sol	TNODE 0x98386F210af731ECbeE7cbbA12C47A8E65bC8856 TNODE BUSD PCS-LP 0x44dC7FE8e51076De1B9f863138107148b441853C
staking-vault-contracts/ Migrations.sol	0x09bF91eA8158E61116eDc0C79e6150981e81D88f 0x14b7B9e0c63a1360315b15AD5eD6Ba681eeDa836
tnode-token-contract/ Migrations.sol	N/A
tnode-token-contract/ Token.sol	0x0E95B13539D0381AB20B4E2893E926Fc99b3d8Dc

Revisions

Revision 1:

- staking-vault-contracts 1cd8bd23210e4c58025b18cc8fac51c67410b3a
- tnode-token-contract: b9bb06608365d166b66293be1eb83e358a6e6952

Revision 2:

- staking-vault-contracts e3e4dd9ada190ce54ec8a1bf2183244b2be041f5

Revision 3:

- staking-vault-contracts f56557d2109cea240ed5492ff056024dcadd82ce

Revision 4:

- staking-vault-contracts 31872d75d143f08fbabc34bb784cc43760d190c4

Revision 5:

- staking-vault-contracts 20a828224069159d3d6c69a1d90013d5add3d8d7

Revision 6:

- staking-vault-contracts bacd2228ddf6506d2798644c28051f1139b20d22

Revision 7:

- staking-vault-contracts e46edb43cae6a92715aca671adb3431ba8b435c7

Imported Contracts

OpenZeppelin: 4.3.1

Externally Owned Accounts

0x329930b94461f8ccd24751c75ccb5048df69bd92 - owner

Appendix B - Risk Ratings

Risk	Description
High Risk	A fatal vulnerability that can cause the loss of all Tokens / Funds.
Medium Risk	A vulnerability that can cause the loss of some Tokens / Funds.
Low Risk	A vulnerability that can cause the loss of protocol functionality.
Informational	Non-security issues such as functionality, style, and/or convention.

Appendix C - Finding Statuses

Closed	Contracts were modified to permanently resolve the finding.
Mitigated	The finding was resolved by other methods such as revoking contract ownership. The issue may require monitoring, for example in the case of a time lock.
Partially Closed	Contracts were updated to fix the issue in some parts of the code.
Partially Mitigated	Fixed by project-specific methods which cannot be verified on-chain. Examples include compounding at a given frequency.
Open	The finding was not addressed.

Appendix D - Testing Standard

An ordinary audit is conducted using these steps.

1. Gather all information
2. Conduct a first visual inspection of documents and contracts
3. Go through all functions of the contract manually (2 independent auditors)
    a. Discuss findings
4. Use specialized tools to find security flaws
    a. Discuss findings
5. Follow up with project lead of findings
6. If there are flaws, and they are corrected, restart from step 2
7. Write and publish a report

During our audit, a thorough investigation has been conducted employing both automated analysis and manual inspection techniques. Our auditing method lays a particular focus on the following important concepts:

- Ensuring that the code and codebase use best practices, industry standards, and available libraries.
- Testing the contract from different angles ensuring that it works under a multitude of circumstances.
- Analyzing the contracts through databases of common security flaws.

Follow Obelisk Auditing for the Latest Information

 ObeliskOrg

 ObeliskOrg

OBELISK

Part of Tibereum Group