OBELISK

Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 38	2022-09-10	Plemonade, Donut	Audit Final

Audit Notes

Audit Date	2022-08-24 - 2022-09-09
Auditor/Auditors	Plemonade, ByFixter
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	OB565658741

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. In instances where an auditor or team member has a personal connection with the audited project, that auditor or team member will be excluded from viewing or impacting any internal communication regarding the specific audit.

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	Based
Description	BASED Next Generation protocol is the first pegless seigniorage protocol exploring DeFi on the FTM Network. We are introducing innovative yield strategies whilst providing inclusivity for Based Finance (V1), that successfully finished emissions. Based Next Gen is a multi-token protocol which consists of the following tokens: $OBOL - PEGLESS token with elastic supply. $SMELT - protocol's underlying Perpetual Print (PP) mechanism emitting OBOL.
Website	https://basedfinance.io/
Contact	https://twitter.com/BasedFinance_io
Contact information	@@athena_goddazz on TG
Token Name(s)	N/A
Token Short	N/A
Contract(s)	See Appendix A
Code Language	Solidity
Chain	Fantom

Audit of Based 3

Obelisk was commissioned by Based on the 22nd of August 2022 to conduct a comprehensive audit of Baseds' additional contracts. The following audit was conducted between the 24th of August 2022 and the 9th of September 2022. 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.

The contracts for this audit are closed source so the audit report includes no code snippets. The contracts will not be verified on-chain, but we have confirmed that the deployed bytecode matches the audited contracts for the on-chain portion of the audit.

During the audit, we found multiple instances of Low, Medium, and High-Risk issues. These issues were conveyed to the project team which worked to solve most of them. Issue #3 which is a High-Risk issue is partially closed as it needs a timelock in order for the user to react to changes. Similar to issue #15 which is still open because of a missing timelock. Issue #9 is a Low-Risk issue that is still open and which needs the user to read the comment on the issue in order to understand its concern. All other findings of relevance are solved in the contracts.

When it comes to the on-chain analysis, issue #17 is open until a timelock is implemented and which then also closes issues #3 and #15. Issue #18 relates to the deployed contracts that are not verified on-chain which means that no one outside of those that have the original code can check on the code of the deployed contracts, which makes sense since the code is closed-source but still warrants an open issue.

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, however, please take note of them.

The team has not reviewed the UI/UX, logic, team, or tokenomics of the Based project**.**

This document is a summary of the findings that the auditors found. Please read the full document for a complete understanding of the audit.

Summary Table

Code Analysis

Finding	ID	Severity	Status
Operator Can Transfer All USDC	#0001	High Risk	Closed
No Protection Against Insufficient Funds	#0002	High Risk	Closed
External Dependencies	#0003	High Risk	Partially Closed
No Protection From Sandwich Attacks	#0004	Medium Risk	Closed
Contract Might Underflow And Revert	#0005	Medium Risk	Closed
Fetched WETH Address Can Be Freeze Rewards	#0006	Medium Risk	Closed
Token Decimals Are Assumed	#0007	Low Risk	Closed
Tokens With Transfer Fee Not Supported	#0008	Low Risk	Closed
Estimate Swap Estimates Price After Swap	#0009	Low Risk	Open
Unnecessary Code	#0010	Informational	Closed
Underflow Causing Locked Assets	#0011	High Risk	Closed
Transfer Fee Can Freeze Assets	#0012	High Risk	Closed
Division Before Multiplication	#0013	High Risk	Closed
Rounding Error Reduces Rewards	#0014	Informational	Open
Oracle Can Be Changed	#0015	Low Risk	Open
Retiring Nodes Requires User Interaction	#0016	Informational	Open

On-Chain Analysis

Finding	ID	Severity	Status
No Timelock	#0017	High Risk	Open
Swapper Contract Is Unverified	#0018	Medium Risk	Open

Findings

Code Analysis

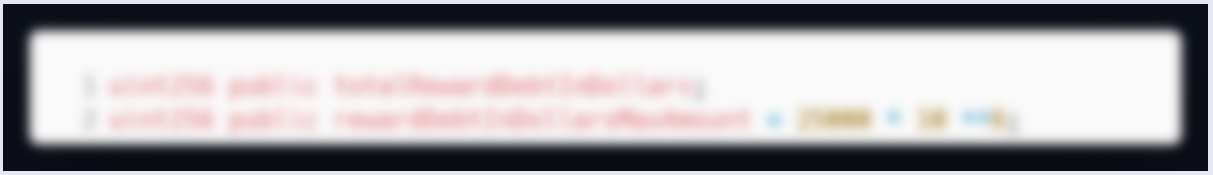
Operator Can Transfer All USDC

FINDING ID	#0001
SEVERITY	High Risk
STATUS	Closed
LOCATION	TwistedNodes.sol -> 106-108

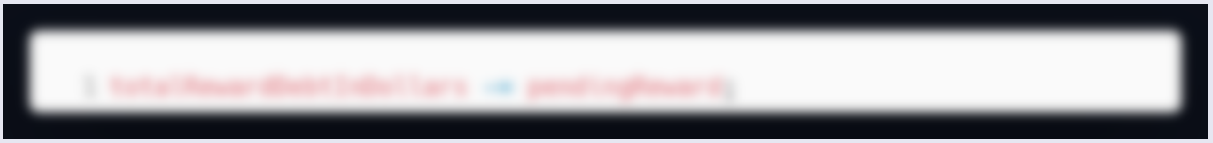
DESCRIPTION	The operator is able to transfer all the USDC out of the contract. Currently, the contract is not aware of how many funds are needed to fully pay all the users risk free so transferring USDC out of the contract could mean that the remaining amount is insufficient to pay all users.
RECOMMENDATION	Remove this function. Another way would be to have a minimum amount of USDC in the contract and allow excess funds to be removed.
RESOLUTION	The project team has removed the function.

No Protection Against Insufficient Funds

FINDING ID	#0002
SEVERITY	High Risk
STATUS	Closed
LOCATION	TwistedNodes.sol -> 32-34



LOCATION	TwistedNodes.sol -> 259-259



DESCRIPTION	There are no guarantees that the contract will have USDC funds to pay out any rewards. Even if there are initial funds to pay the current debt, as USDC is paid out, the total debt is decreased (line 259) and new investors can invest and therefore require a steady external supply of USDC. This contract needs an external supplier of USDC. There is no guarantee that any was USDC supplied.
RECOMMENDATION	Add a mechanism to ensure that sufficient USDC is available.
RESOLUTION	Project team implemented the recommended fix.

External Dependencies

FINDING ID	#0003
SEVERITY	High Risk
STATUS	Partially Closed
LOCATION	TwistedNodes.sol -> 40-44

LOCATION	TwistedNodes.sol -> 64-74

DESCRIPTION	The swapper contract and oracle contract is currently unknown and obelisk has not received these contracts. Conventional behavior is assumed but not guaranteed. Unexpected behavior could be used to steal rewards and deposits.
RECOMMENDATION	Ensure that 3rd party contracts behave correctly.
RESOLUTION	The external dependencies have been looked at and currently, they need to be locked down as users' deposits are locked for a long time. An alternative is making the user able to change the swapper and oracle themselves in case migration is needed.

External contracts can be changed without delay, see finding #17.

Project team comment: *“we will be adding timelocks in the future to the contract.”*

No Protection From Sandwich Attacks

FINDING ID	#0004
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	TwistedNodes.sol -> 225-278

DESCRIPTION	The estimate and swap of the rewards happen at transaction time. Thus the price of tokens is directly related to the current price. A malicious actor could front-run big reward claims and manipulate the pool in a sandwich attack.
RECOMMENDATION	Allow the user to input the current value of an estimated swap when sending the transaction and or use an oracle to acquire a fairly accurate price.
RESOLUTION	A min amount parameter was added to add protection against sandwich attacks.

Contract Might Underflow And Revert

FINDING ID	#0005
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	TwistedNodes.sol -> 336-356

DESCRIPTION	If *rewardPaiedInDollars* or *rewardPaiedInToken* is more than *riskFreeValueUSDC* then the contract will revert. If it is 0, then it would also have been 0 in the calculation. Thus these two if statements are unnecessary when solidity 0.8 is used. For example: Suppose a user claimed 9/10 of their *riskFreeValueUSDC* in tokens and then the token value increases so that the claimed amount is worth more than the *riskFreeValueUSDC*. The contract will revert until the

	value drops enough for another withdrawal.
RECOMMENDATION	Instead calculate *rewardPaiedInDollars* > *riskFreeValueUSDC* and the equivalent *rewardPaidInToken* > *riskFreeValueUSDC*.
RESOLUTION	The project team implemented the recommended fix.

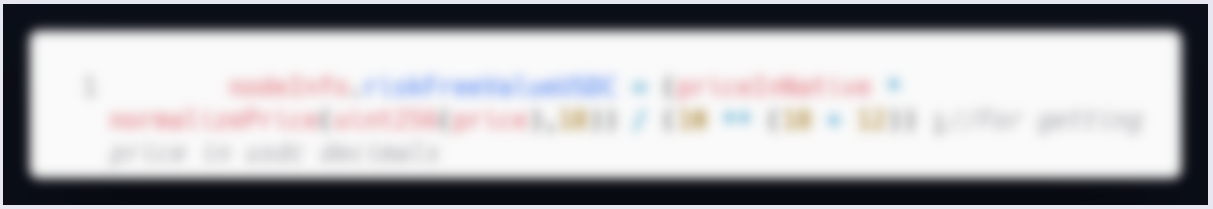
Fetched WETH Address Can Be Freeze Rewards

FINDING ID	#0006
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	TwistedNodes.sol -> 267-271

DESCRIPTION	The swapper contract can change the address of WETH and force a revert or other behavior when calling *withdraw*. Even if the swapper is static the operator can disable the reward token and force the use of a new reward token with a different swapper contract.
RECOMMENDATION	Set a constant or immutable WETH address at compile time.
RESOLUTION	The project team added a native wrapped token as a constant variable.

Token Decimals Are Assumed

FINDING ID	#0007
SEVERITY	Low Risk
STATUS	Closed
LOCATION	TwistedNodes.sol -> 173

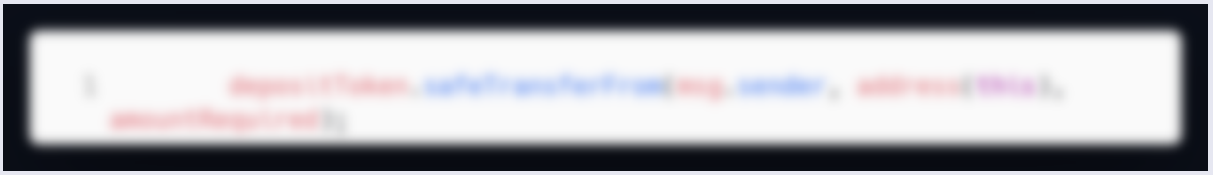


LOCATION	TwistedNodes.sol -> 359-375

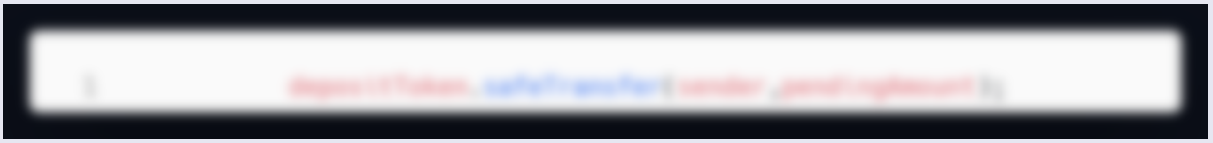
DESCRIPTION	Token prices are normalized to use hard-coded decimal places. This can lead to unexpected calculation results in tokens that use a different number of decimals.
RECOMMENDATION	Remove the hard-coded assumptions and query tokens for their decimals.
RESOLUTION	The project team now fetches the decimals. The fetching of decimals can be done once in the constructor and stored as an immutable variable.

Tokens With Transfer Fee Not Supported

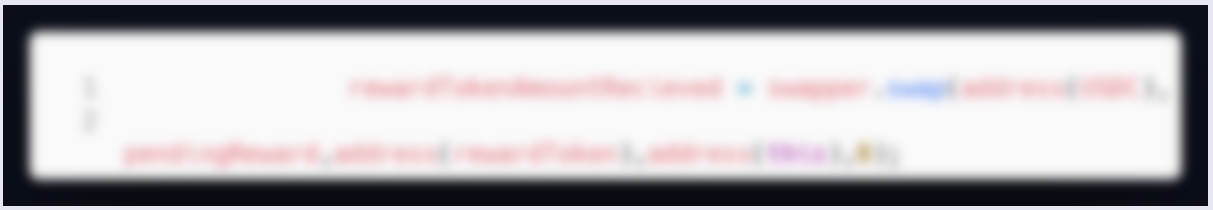
FINDING ID	#0008
SEVERITY	Low Risk
STATUS	Closed
LOCATION	TwistedNodes.sol -> 157



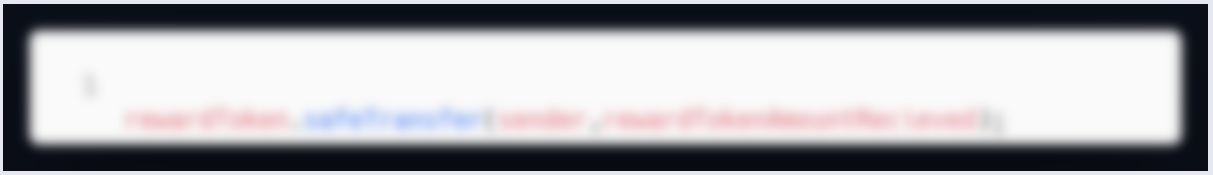
LOCATION	TwistedNodes.sol -> 210



LOCATION	TwistedNodes.sol -> 253-254



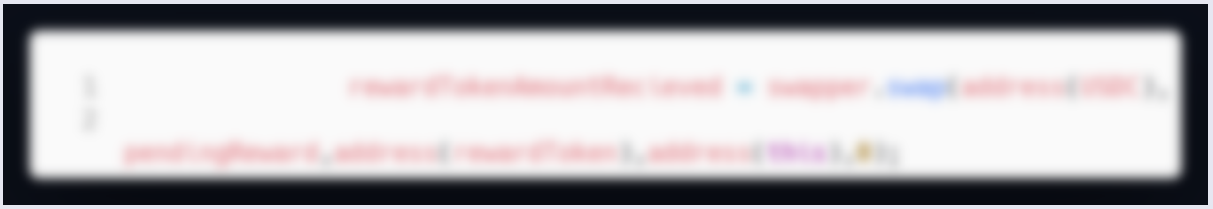
LOCATION	TwistedNodes.sol -> 274



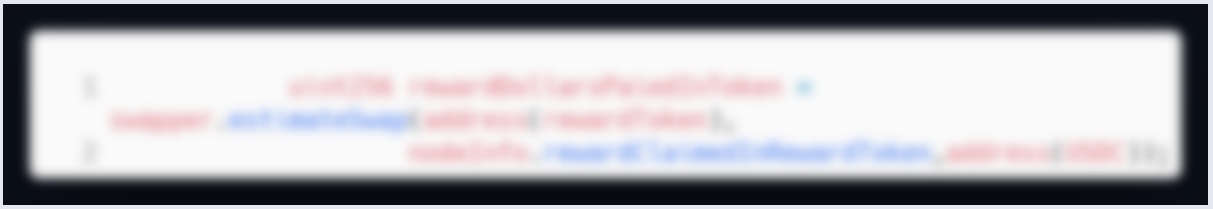
DESCRIPTION	The contract does not support tokens with a transfer fee as it assumes all transfers to transfer 100% of tokens.
RECOMMENDATION	Make sure tokens with transfer fees aren't added.
RESOLUTION	The contract was modified to allow for the withdrawal of a transfer fee token.

Estimate Swap Estimates Price After Swap

FINDING ID	#0009
SEVERITY	Low Risk
STATUS	Open
LOCATION	TwistedNodes.sol -> 253-254



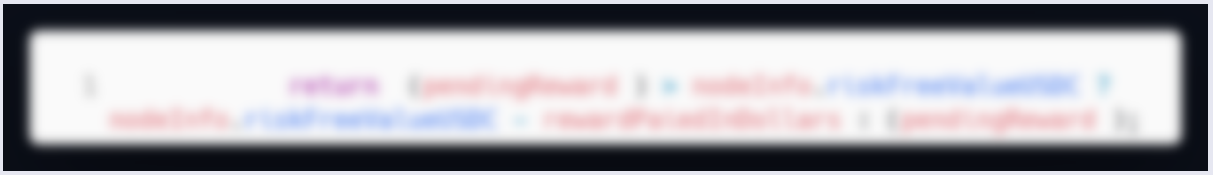
LOCATION	TwistedNodes.sol -> 261-262



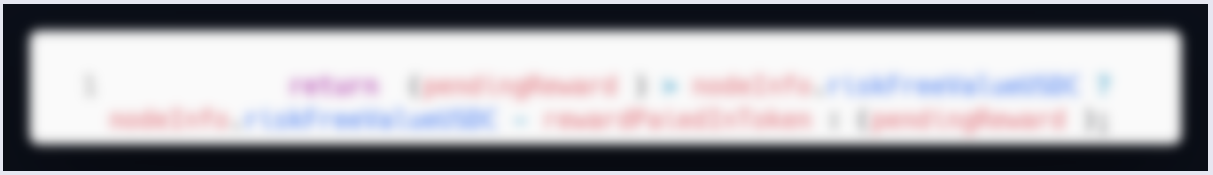
DESCRIPTION	After a swap is made the price will have changed. Thus the node owner will receive just a bit less because of the previous swap(in USD). For a low-volume token, this difference will be greater.
RECOMMENDATION	Instead of estimating after the swap, use the amount swapped.
RESOLUTION	N/A

Unnecessary Code

FINDING ID	#0010
SEVERITY	Informational
STATUS	Closed
LOCATION	TwistedNodes.sol -> 326



LOCATION	TwistedNodes.sol -> 332



DESCRIPTION	The noted return statements are in effect a min of *pendingReward* and *riskFreeValueUSDC*. Note that in the current implementation, *pendingReward* will always be the lesser of the values. *claimPerSec* is the total reward time so at most, *pendingReward* should be the same as *riskFreeValueUSDC*.
RECOMMENDATION	Simplify the code to reduce complexity and the possibility of calculation errors.
RESOLUTION	Project team has removed the code.

Underflow Causing Locked Assets

FINDING ID	#0011
SEVERITY	High Risk
STATUS	Closed
LOCATION	rev2 - TwistedNodes.sol -> 256-265

DESCRIPTION	*totalRewardDebtInDollars* is decremented twice: First with the pendingReward and then again for the whole amount when the node is retired. *pendingReward* should be the only thing to decrement. When enough users have retired their nodes the claiming will revert as *totalRewardDebtInDollars* will underflow. Because the claiming as a node is retired has 2x the impact on *totalRewardDebtInDollars* the forever locked funds amount will increase as more people deposit and claim rewards.
RECOMMENDATION	Use *nodeInfo.rewardClaimedInDollars* and *nodeInfo.riskFreeValueUSDC* to remove the remaining on *totalRewardDebtInDollars* as the node is retired.
RESOLUTION	The project team has implemented the recommendation.

Transfer Fee Can Freeze Assets

FINDING ID	#0012
SEVERITY	High Risk
STATUS	Closed
LOCATION	rev2 - TwistedNodes.sol -> 201-203

LOCATION	rev2 - TwistedNodes.sol -> 246-250

LOCATION	rev2 - TwistedNodes.sol -> 273-275

DESCRIPTION	If a transfer fee token is deposited or used as a rewards token, then all attempts to claim rewards will fail as the require statement will not be satisfied. This could occur as well if a transfer fee is enabled after a user deposits.

RECOMMENDATION	Remove the checks on the claim functions or add a parameter that bypasses the *require* statement on the claim functions.
RESOLUTION	N/A

Division Before Multiplication

FINDING ID	#0013
SEVERITY	High Risk
STATUS	Closed
LOCATION	rev3 - TwistedNodes.sol -> 316-327

DESCRIPTION	The calculations noted use mixed orders of multiplication and division. *claimPerSec* in some cases will round down to 0. However, the users would be able to claim after the whole node lifespan.
RECOMMENDATION	Change the calculations to first multiply, then divide.
RESOLUTION	The team fixed the order of operations. Users might earn a bit less within the lifespan of the node. See #14.

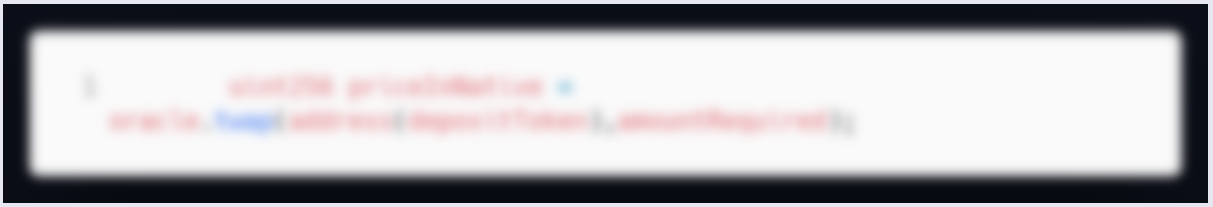
Rounding Error Reduces Rewards

FINDING ID	#0014
SEVERITY	Informational
STATUS	Open
LOCATION	rev4 - TwistedNodes.sol -> 316-327

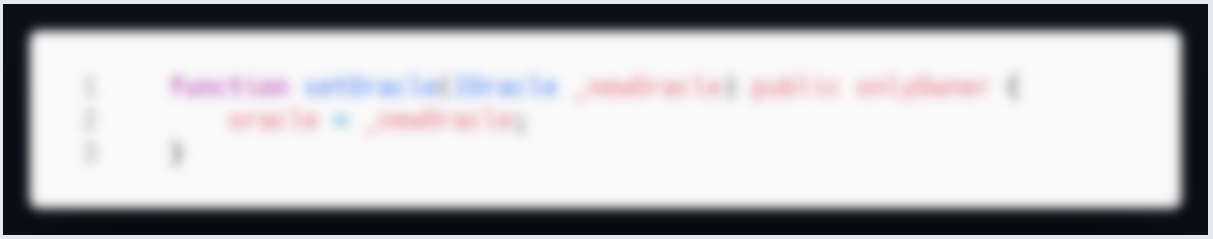
DESCRIPTION	The reward calculation causes a rounding error which leaves some tokens undistributed. However, the users would be able to claim after the whole node lifespan.
RECOMMENDATION	Fix the rounding error to ensure tokens are correctly distributed.
RESOLUTION	N/A

Oracle Can Be Changed

FINDING ID	#0015
SEVERITY	Low Risk
STATUS	Open
LOCATION	rev-5 - TwistedNodes.sol -> 169



LOCATION	rev-5 - TwistedNodes.sol -> 402-405



DESCRIPTION	On a deposit the *oracle.twap* is called to check the price of the deposit token. If a malicious actor has the Owner role, they could frontrun a deposit and switch out the oracle to a malicious one. This could cause the depositor to lose their funds.
RECOMMENDATION	Add a *require* statement with a minimum amount of input from the user. Alternatively, add a timelock to safeguard against manipulation of the oracle.
RESOLUTION	No timelock was added to the deployed contract. See finding #17.

Rounding Error Reduces Rewards

FINDING ID	#0016
SEVERITY	Informational
STATUS	Open
LOCATION	rev-5 - TwistedNodes.sol -> 227-282

DESCRIPTION	A Node is only retired after a claim is called. This can become an issue when previously paid tokens have gone up in price. In such a case, a user has no incentive to claim and retire their node. Because the total debt of the contract won't decrease if the token ever goes down in price they could claim the rest of the unclaimed USDC amount. This could give certain users an unintended advantage.
RECOMMENDATION	Provide contract information to all users to remove unintended advantages.
RESOLUTION	N/A

On-Chain Analysis

No Timelock

FINDING ID	#0017
SEVERITY	High Risk
STATUS	Open
LOCATION	MEDIUM NODES: 0x525ca3877a78c6AE12292D0a55765775e3943379 SMELT - WFTM Uniswap Oracle 0x5E92ccb0Bd8071B9793f23eB4D7F3b818aE59536

DESCRIPTION	The noted contracts have not had their ownership transferred to the timelock contract. A malicious actor as the contract owner can change the chainlink price feed and oracle addresses. This can be used to cause deposits to be recorded as zero, or to prevent the withdrawal of any rewards. Additionally, the connected uniswap oracle does not have a timelock, which can allow its period to be manipulated to use out-of-date prices. This can have a similar but reduced effect.
RECOMMENDATION	Add a timelock to the contracts.
RESOLUTION	Project team comment: *“we will be adding timelocks in the future to the contract.”*

Swapper Contract Is Unverified

FINDING ID	#0018
SEVERITY	Medium Risk
STATUS	Open
LOCATION	Swapper 0x63A8e53DbD8fbf08DfA9A92d550468DaF62f65b1

DESCRIPTION	The swapper contract was not verified on chain. As a result, its functionality could not be confirmed. This contract is used to convert all rewards to the user's specified reward token. If the swapper contract is attacked, or malfunctions, it can lead to the loss of expected rewards.
RECOMMENDATION	Verify the contract on the chain.
RESOLUTION	N/A

External Addresses

Externally Owned Accounts

Owner/Operator

ACCOUNT	0xB9B22504b9071291E938E0E582934A82C4a4670c
USAGE	0x525ca3877a78c6AE12292D0a55765775e3943379 *TwistedNodes.owner* - Variable *TwistedNodes.operator* - Variable
IMPACT	• receives elevated permissions as owner, operator, or other

External Contracts

These contracts are not part of the audit scope.

SMELT - WFTM Uniswap Oracle

ADDRESS	0x5E92ccb0Bd8071B9793f23eB4D7F3b818aE59536
USAGE	0x525ca3877a78c6AE12292D0a55765775e3943379 *TwistedNodes.oracle* - Variable
IMPACT	• Uniswap Oracle • impacts ability to deposit or withdraw tokens

FTM / USD Chainlink price feed

ADDRESS	0xf4766552d15ae4d256ad41b6cf2933482b0680dc
USAGE	0x525ca3877a78c6AE12292D0a55765775e3943379 *TwistedNodes.chainLinkAggregator* - Variable
IMPACT	• ERC20 Token • impacts ability to deposit or withdraw tokens

Swapper

ADDRESS	0x63A8e53DbD8fbf08DfA9A92d550468DaF62f65b1
USAGE	0x525ca3877a78c6AE12292D0a55765775e3943379 *TwistedNodes.rewardTokens[x].swapper* - Variable (no setter)
IMPACT	• receives transfer of tokens deposited or minted by project

External Tokens

These contracts are not part of the audit scope.

SMELT

ADDRESS	0x141FaA507855E56396EAdBD25EC82656755CD61e
USAGE	0x525ca3877a78c6AE12292D0a55765775e3943379 *TwistedNodes.depositToken* - Variable (no setter) *TwistedNodes.rewardTokens[2].token* - Variable (no setter)
IMPACT	• ERC20 Token

USDC

ADDRESS	0x04068DA6C83AFCFA0e13ba15A6696662335D5B75
USAGE	0x525ca3877a78c6AE12292D0a55765775e3943379 *TwistedNodes.USDC* - Variable (no setter) *TwistedNodes.rewardTokens[1].token* - Variable (no setter)
IMPACT	• ERC20 Token

Wrapped Ether

ADDRESS	0x21be370D5312f44cB42ce377BC9b8a0cEF1A4C83
USAGE	0x525ca3877a78c6AE12292D0a55765775e3943379 *TwistedNodes.WETH* - Constant *TwistedNodes.rewardTokens[0].token* - Variable (no setter)
IMPACT	• ERC20 Token

Appendix A - Reviewed Documents

Deployed Contracts

Document	Address
TwistedNodes	0x525ca3877a78c6AE12292D0a55765775e3943379

Libraries And Interfaces

interfaces/IOracle.sol interfaces/ISwapper.sol interfaces/IUniswapV2Router.sol interfaces/IWeth.sol lib/TransferHelper.sol owner/TwistedNodes.sol

Revisions

Revision 1	d94aa38596305a189b62cedf54077ca7ff930625
Revision 2	9e39284ad51073f6d16e5981ee4696f3bd6cb7c2
Revision 3	7388ff3c6b678db4371af6165b43ee6248415814
Revision 4	9fa4534f9263537c768667a1cf0e3512bf3c84b5
Revision 5	d940b6caf30072a61a7da0352aa8e27c712255ec

Imported Contracts

OpenZeppelin	4.5.0
Chainlink	0.4.0

Appendix B - Risk Ratings

Risk	Description
High Risk	Security risks that are ***almost certain*** to lead to ***impairment or loss of funds***. Projects are advised to fix as soon as possible.
Medium Risk	Security risks that are ***very likely*** to lead to ***impairment or loss of funds*** with ***limited impact***. Projects are advised to fix as soon as possible.
Low Risk	Security risks that can lead to ***damage to the protocol***. Projects are advised to fix. Issues with this rating might be used in an exploit with other issues to cause significant damage.
Informational	Noteworthy information. Issues may include code conventions, missing or conflicting information, gas optimizations, and other advisories.

Appendix C - Finding Statuses

Closed	Contracts were modified to permanently resolve the finding.
Mitigated	The finding was resolved on-chain. The issue may require monitoring, for example in the case of a time lock.
Partially Closed	Contracts were modified to partially fix the issue
Partially Mitigated	The finding was resolved by project specific methods which cannot be verified on chain. Examples include compounding at a given frequency, or the use of a multisig wallet.
Open	The finding was not addressed.

Appendix D - Glossary

Contract Structure

Contract: An address with which provides functionality to users and other contracts. They are implemented in code and deployed to the blockchain.
Protocol: A system of contracts which work together.
Stakeholders: The users, operators, owners, and other participants of a contract.

Security Concepts

Bug**:** A defect in the contract code.
Exploit**:** A chain of events involving bugs, vulnerabilities, or other security risks which damages a protocol.
Funds: Tokens deposited by users or other stakeholders into a protocol.
Impairment: The loss of functionality in a contract or protocol.
Security risk: A circumstance that may result in harm to the stakeholders of a protocol. Examples include vulnerabilities in the code, bugs, excessive permissions, missing timelock, etc.
Vulnerability: A vulnerability is a flaw that allows an attacker to potentially cause harm to the stakeholders of a contract. They may occur in a contract's code, design, or deployed state on the blockchain.

Appendix E - Audit Procedure

A typical Obelisk audit uses a combination of the three following methods:

Manual analysis consists of a direct inspection of the contracts to identify any security issues. Obelisk auditors use their experience in software development to spot vulnerabilities. Their familiarity with common contracts allows them to identify a wide range of issues in both forked contracts as well as original code.

Static analysis is software analysis of the contracts. Such analysis is called “static” as it examines the code outside of a runtime environment. Static analysis is a powerful tool used by auditors to identify subtle issues and to verify the results of manual analysis.

On-chain analysis is the audit of the contracts as they are deployed on the block-chain. This procedure verifies that:

- deployed contracts match those which were audited in manual/static analysis;
- contract values are set to reasonable values;
- contracts are connected so that interdependent contract function correctly;
- and the ability to modify contract values is restricted via a timelock or DAO mechanism. (We recommend a timelock value of at least 72 hours)

Each obelisk audit is performed by at least two independent auditors who perform their analysis separately.

After the analysis is complete, the auditors will make recommendations for each issue based on best practice and industry standards. The project team can then resolve the issues, and the auditors will verify that the issues have been resolved with no new issues introduced.

Our auditing method lays a particular focus on the following important concepts:

- Quality code and the use of best practices, industry standards, and thoroughly tested libraries.
- Testing the contract from different angles to ensure that it works under a multitude of circumstances.
- Referencing the contracts through databases of common security flaws.

Follow Obelisk Auditing for the Latest Information

 ObeliskOrg

 ObeliskOrg

OBELISK

Part of Tibereum Group