OBELISK

Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 36	2022-03-02	Donut, DoD4uFN	Audit Final

Audit Notes

Audit Date	2022-02-10 - 2022-03-02
Auditor/Auditors	DoD4uFN, 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	OB585655238

Disclaimer

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	Ripae Finance
Description	Ripae Finance's full focus is to build a true cross-chain algorithmic stable coin protocol that is stabilized with true use-cases all around the DeFi Ecosystem.
Website	https://ripae.finance/
Contact	https://twitter.com/ripaefinance
Contact information	RPD#6260 on Discord
Token Name(s)	N/A
Token Short	N/A
Contract(s)	See Appendix A
Code Language	Solidity
Chain	Fantom, Avalanche

Audit of Ripae Finance

Obelisk was commissioned by Ripae Finance on the 9th of February 2022 to conduct a comprehensive audit of Ripaes' contracts. The following audit was conducted between the 10th of February 2022 and the 2nd of March 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 on-chain analysis was conducted on Avalanche Deployment. Overall there were a number of medium and low-risk issues found. All Medium-Risk issues were closed besides issue #3 which is partially mitigated.

When it comes to Low-Risk issues, #8, #10, #15, #16 are currently open and acknowledged by the project team. Issue #8 refers to variables with missing fixed lower and upper limits. Issue #10 refers to the lack of a SafeTransfer function that can catch transfer errors and have nothing to do with the user funds. Issue #15 and issue #16 refers to a missing timelock to EOA address.

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 Ripae 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

Finding	ID	Severity	Status
No Emergency Withdraw Functionality	#0001	Medium Risk	Closed
Pae And LPs Are Withdrawable 90 Days After The End Of Pool	#0002	Medium Risk	Closed
Distribute Reward Does Not Check Destination Address	#0003	Medium Risk	Partially Mitigated
Require Statement Checks Parameter Instead Of Variable	#0004	Medium Risk	Closed
Timelock Minimum Delay Is Short	#0005	Low Risk	Mitigated
Lockup Period Of Funds Can Be Increased Or Decreased	#0006	Low Risk	Mitigated
Unbounded Loop	#0007	Informational	Open
No Limit For Protocol Values	#0008	Low Risk	Open
Redundant Require Statement	#0009	Informational	Open
Use Safe Transfer	#0010	Low Risk	Open
Contract Values Can Be Immutable Or Constant (Gas Optimization)	#0011	Informational	Open
Missing Zero Checks	#0012	Informational	Open
Unused Imports	#0013	Informational	Open
No Events Emitted For Changes To Protocol Values	#0014	Informational	Open
Operator Is An EOA	#0015	Low Risk	Open
TreasuryFund Is An EOA	#0016	Low Risk	Open

Findings

Manual Analysis

No Emergency Withdraw Functionality

FINDING ID	#0001
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	Masonry.sol

DESCRIPTION	The protocol does not have an emergency withdrawal function.
RECOMMENDATION	Add functionality that allows users to withdraw their funds and abandon their rewards. (after their lockup period)
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 3a5f5462f83cd77dba35a459ca42c8724a779c7e

Pae And LPs Are Withdrawable 90 Days After The End Of Pool

FINDING ID	#0002
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	[distribution/GenesisRewardPool.sol -> 259-270]()

```
    function governanceRecoverUnsupported(IERC20 _token, uint256 amount, address to) external onlyOperator {
        if (block.timestamp < poolEndTime + 90 days) {
            // do not allow to drain core token (Reward or lps) if less than 90 days after pool ends
            require(_token != reward, "reward");
            uint256 length = poolInfo.length;
            for (uint256 pid = 0; pid < length; ++pid) {
                PoolInfo storage pool = poolInfo[pid];
                require(_token != pool.token, "pool.token");
            }
        }
        _token.safeTransfer(to, amount);
    }
```

LOCATION	[distribution/PaeRewardPool.sol -> 260-271]()

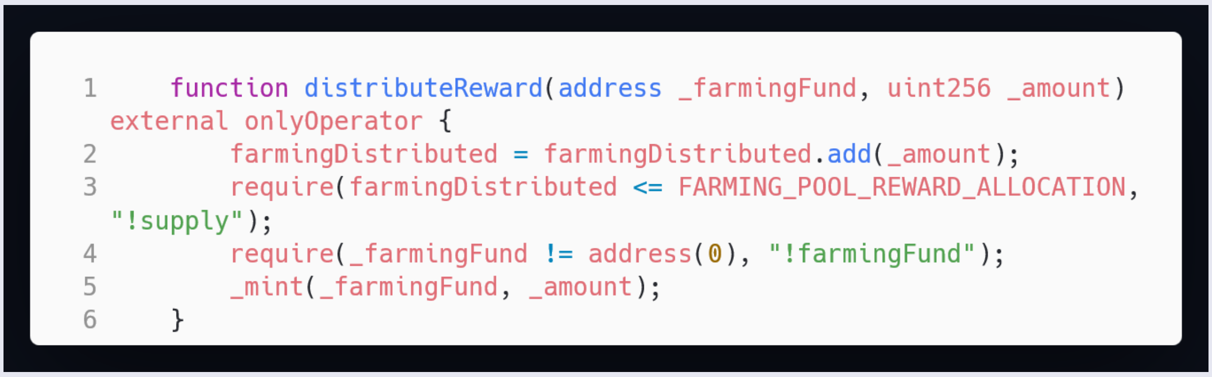
```
    function governanceRecoverUnsupported(IERC20 _token, uint256 amount, address to) external onlyOperator {
        if (block.timestamp < poolEndTime + 90 days) {
            // do not allow to drain core token (PAE or lps) if less than 90 days after pool ends
            require(_token != pae, "pae");
            uint256 length = poolInfo.length;
            for (uint256 pid = 0; pid < length; ++pid) {
                PoolInfo storage pool = poolInfo[pid];
                require(_token != pool.token, "pool.token");
            }
        }
        _token.safeTransfer(to, amount);
    }
```

DESCRIPTION	The Reward Pool contract has a function to recover tokens that have been mistakenly sent to the contract, that is callable only by the owner. In order to prevent the owner from withdrawing LPs or the project's token, a condition has been added to the

	function. That condition is ignored 90 days after the end of the pool.
RECOMMENDATION	Do not allow the withdrawal of LPs or the project's token.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 3a5f5462f83cd77dba35a459ca42c8724a779c7e

Distribute Reward Does Not Check Destination Address

FINDING ID	#0003
SEVERITY	Medium Risk
STATUS	Partially Mitigated
LOCATION	Pae.sol -> 102-107

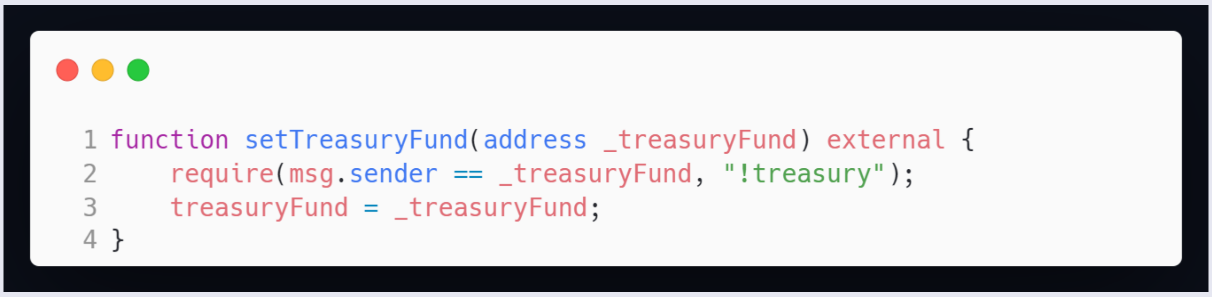


```
    function distributeReward(address _farmingFund, uint256 _amount)
external onlyOperator {
        farmingDistributed = farmingDistributed.add(_amount);
        require(farmingDistributed <= FARMING_POOL_REWARD_ALLOCATION,
"!supply");
        require(_farmingFund != address(0), "!farmingFund");
        _mint(_farmingFund, _amount);
    }
```

DESCRIPTION	The *distributeReward* function does not properly check the *_farmingFund* address before it mints tokens to it.
RECOMMENDATION	Ensure that *_farmingFund* address is the intended farm.
RESOLUTION	Rewards will be distributed to other chains as well, which implies that funds will be moved to a non-farm address. At the on-chain analysis Obelisk verified that the owner of Pae.sol on Avalanche is the *AnyswapV4Router*. The Pae.sol on Fantom has no owner, and it’s operator is a timelock of 6 hours. Reviewed in commit a5c2004bbdae37c3c0533994d9ec2dee558e34f8

Require Statement Checks Parameter Instead Of Variable

FINDING ID	#0004
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	[Pae.sol -> 54-57]()



```
function setTreasuryFund(address _treasuryFund) external {
    require(msg.sender == _treasuryFund, "!treasury");
    treasuryFund = _treasuryFund;
}
```

DESCRIPTION	Require statement is comparing the *msg.sender* with the function parameter. This would allow any user to become the treasuryFund.
RECOMMENDATION	Change the statement to compare *msg.sender* to *treasuryFund* instead of *_treasuryFund* address.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit a5c2004bbdae37c3c0533994d9ec2dee558e34f8

Timelock Minimum Delay Is Short

FINDING ID	#0005
SEVERITY	Low Risk
STATUS	Mitigated
LOCATION	Timelock.sol -> 16

```
uint public constant MINIMUM_DELAY = 6 hours;
```

DESCRIPTION	The timelock delay is set to 6 hours.
RECOMMENDATION	Obelisk recommends timelocks of at least 72 hours.
RESOLUTION	The timelock was deployed with 24 hours minimum delay instead of 6 hours. Reviewed in commit 3a5f5462f83cd77dba35a459ca42c8724a779c7e

Lockup Period Of Funds Can Be Increased Or Decreased

FINDING ID	#0006
SEVERITY	Low Risk
STATUS	Mitigated
LOCATION	Masonry.sol -> 141-145

```
    function setLockUp(uint256 _withdrawLockupEpochs, uint256 _rewardLockupEpochs) external onlyOperator {
        require(_withdrawLockupEpochs >= _rewardLockupEpochs && _withdrawLockupEpochs <= 56, "_withdrawLockupEpochs: out of range"); // <= 2 week
        withdrawLockupEpochs = _withdrawLockupEpochs;
        rewardLockupEpochs = _rewardLockupEpochs;
    }
```

LOCATION	Masonry.sol -> 211

```
require(masons[msg.sender].epochTimerStart.add(withdrawLockupEpochs) <= treasury.epoch(), "Masonry: still in withdraw lockup");
```

LOCATION	Masonry.sol -> 224

```
require(masons[msg.sender].epochTimerStart.add(rewardLockupEpochs) <= treasury.epoch(), "Masonry: still in reward lockup");
```

DESCRIPTION	Users can experience increased or decreased lockup periods because the *Masonseat* struct is storing the *epochTimerStart*. If the *withdrawLockupEpochs* is increased or decreased before they are able to withdraw their funds, they will face an increased or decreased lockup period accordingly. Note: The maximum lockup period is 2 weeks.
RECOMMENDATION	Instead of storing the epoch which the user staked, store the epoch which the user will be able to withdraw/claim their rewards.
RESOLUTION	The project team confirmed that this is the intended

	behavior. Reviewed in commit [3a5f5462f83cd77dba35a459ca42c8724a779c7e](3a5f5462f83cd77dba35a459ca42c8724a779c7e)

Unbounded Loop

FINDING ID	#0007
SEVERITY	Informational
STATUS	Open
LOCATION	• Treasury.sol -> 484-489: *for (uint8 tierId = 8; tierId >= 0; --tierId) {*

DESCRIPTION	The for loop's condition cannot be met. The reason is that the condition will be met when the index *tierId* is a negative number, which cannot happen because it's a *uint8* (unsigned integer). Therefore, the for loop won't necessarily stop at 9 iterations.
RECOMMENDATION	Change the for loop so that it stops after a number of iterations, instead of relying on the if condition.
RESOLUTION	No changes were made. Reviewed in commit a5c2004bbdae37c3c0533994d9ec2dee558e34f8

No Limit For Protocol Values

FINDING ID	#0008
SEVERITY	Low Risk
STATUS	Open
LOCATION	• Treasury.sol -> 358-360: *function setMaxDiscountRate(uint256 _maxDiscountRate) external onlyOperator {* • Treasury.sol -> 362-364: *function setMaxPremiumRate(uint256 _maxPremiumRate) external onlyOperator {*

DESCRIPTION	These protocol variables do not have upper or lower bounds.
RECOMMENDATION	Add appropriate limits to the protocol variables.
RESOLUTION	No changes were made, although the variables *discountPercent* and *premiumPercent* provide a limit to the bond discount and premium rates respectively. Reviewed in commit a5c2004bbdae37c3c0533994d9ec2dee558e34f8

Redundant Require Statement

FINDING ID	#0009
SEVERITY	Informational
STATUS	Open
LOCATION	Treasury.sol -> 300 Treasury.sol -> 313

```
        require(_index >= 0, "Index has to be higher than 0");
```

DESCRIPTION	Parameter *_index* is a *uint8* which is always positive. Additionally, the message of the require statement does not reflect the statement check.
RECOMMENDATION	Remove the require statement.
RESOLUTION	No changes were made. Reviewed in commit a5c2004bbdae37c3c0533994d9ec2dee558e34f8

Static Analysis

Use Safe Transfer

FINDING ID	#0010
SEVERITY	Low Risk
STATUS	Open
LOCATION	• Pae.sol -> 118: *_token.transfer(_to, _amount);* • PegToken.sol -> 39: *_token.transfer(_to, _amount);* • Treasury.sol -> 464: *IERC20(pToken).transfer(daoFund, _daoFundSharedAmount);* • Treasury.sol -> 471: *IERC20(pToken).transfer(devFund, _devFundSharedAmount);*

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	No changes were made, although the aforementioned unchecked transfer calls do not interact with users' funds. Reviewed in commit a5c2004bbdae37c3c0533994d9ec2dee558e34f8

Contract Values Can Be Immutable Or Constant (Gas Optimization)

FINDING ID	#0011
SEVERITY	Informational
STATUS	Open
LOCATION	Pae.sol -> 20-24

```
uint256 public startTime;
uint256 public endTime;

uint256 public treasuryFundRewardRate;
uint256 public devFundRewardRate;
```

LOCATION	distribution/GenesisRewardPool.sol -> 48-49

```
uint256 public rewardPerSecond = 0.135 ether;
uint256 public runningTime = 3 days;
```

LOCATION	distribution/PaeRewardPool.sol -> 48-49

```
uint256 public paePerSecond = 0.005390664637 ether; // 170000 /
(365 days * 24h * 60min * 60s)
uint256 public runningTime = 365 days;
```

DESCRIPTION	Variables that do not change during the operation of a contract can be marked as *immutable* or *constant* to reduce gas costs and improve code readability.
RECOMMENDATION	Mark these variables as *immutable* or *constant* accordingly.
RESOLUTION	No changes were made. Reviewed in commit a5c2004bbdae37c3c0533994d9ec2dee558e34f8

Missing Zero Checks

FINDING ID	#0012
SEVERITY	Informational
STATUS	Open
LOCATION	• distribution/GenesisRewardPool.sol -> 255-257: *function setOperator(address _operator) external onlyOperator* • distribution/PaeRewardPool.sol -> 256-258: *function setOperator(address _operator) external onlyOperator* • Masonry.sol -> 137-140: *function setOperator(address _operator) external onlyOperator* • Pae.sol -> 54-56: *function setTreasuryFund(address _treasuryFund) external* • Treasury.sol -> 229-275: *function initialize(address _peg, address _bond, address _pae, address _oracle, address _masonry, address _genesisPool, uint256 _startTime) public notInitialized* • Treasury.sol -> 281-283: *function setMasonry(address _masonry) external onlyOperator* • Treasury.sol -> 285-287: *function setPegOracle(address _oracle) external onlyOperator*

DESCRIPTION	The aforementioned functions can set addresses to the zero address. Zero addresses may cause incorrect contract behavior.
RECOMMENDATION	Add a check to ensure contract values are never set to an invalid zero address.
RESOLUTION	No changes were made. Reviewed in commit a5c2004bbdae37c3c0533994d9ec2dee558e34f8

Unused Imports

FINDING ID	#0013
SEVERITY	Informational
STATUS	Open
LOCATION	Treasury.sol -> 8

```
import "@openzeppelin/contracts/utils/ReentrancyGuard.sol";
```

DESCRIPTION	The imported library is not being used. It's recommended to be removed to avoid confusion.
RECOMMENDATION	Remove the imports.
RESOLUTION	No changes were made. Reviewed in commit a5c2004bbdae37c3c0533994d9ec2dee558e34f8

No Events Emitted For Changes To Protocol Values

FINDING ID	#0014
SEVERITY	Informational
STATUS	Open
LOCATION	• distribution/GenesisRewardPool.sol -> 80-118: *function add(uint256 _allocPoint, IERC20 _token, bool _withUpdate, uint256 _lastRewardTime) public onlyOperator* • distribution/GenesisRewardPool.sol -> 121-130: *function set(uint256 _pid, uint256 _allocPoint) public onlyOperator* • distribution/GenesisRewardPool.sol -> 255-257: *function setOperator(address _operator) external onlyOperator* • distribution/PaeRewardPool.sol -> 81-119: *function add(uint256 _allocPoint, IERC20 _token, bool _withUpdate, uint256 _lastRewardTime) public onlyOperator* • distribution/PaeRewardPool.sol -> 122-131: *function set(uint256 _pid, uint256 _allocPoint) public onlyOperator* • distribution/PaeRewardPool.sol -> 256-258: *function setOperator(address _operator) external onlyOperator* • utils/Epoch.sol -> 77-80: *function setPeriod(uint256 _period) external onlyOperator* • utils/Epoch.sol -> 82-84: *function setEpoch(uint256 _epoch) external onlyOperator*Treasury.sol -> 277-279: *function setOperator(address _operator) external onlyOperator* • Masonry.sol -> 137-139: *function setOperator(address _operator) external onlyOperator* • Masonry.sol -> 141-145: *function setLockUp(uint256 _withdrawLockupEpochs, uint256 _rewardLockupEpochs) external onlyOperator* • Pae.sol -> 54-57: *function setTreasuryFund(address _treasuryFund) external* • Pae.sol -> 59-63: *function setDevFund(address _devFund) external* • Pae.sol -> 65-67: *function setNotifyDevFund(bool _notifyDefFund) external onlyOperator* • Pae.sol -> 86-100: *function claimRewards() external* • Pae.sol -> 102-107: *function distributeReward(address _farmingFund, uint256 _amount) external onlyOperator*

- Treasury.sol -> 281-283: *function setMasonry(address _masonry) external onlyOperator*
- Treasury.sol -> 285-287: *function setPegOracle(address _oracle) external onlyOperator*
- Treasury.sol -> 289-292: *function setPegPriceCeiling(uint256 _pegPriceCeiling) external onlyOperator*
- Treasury.sol -> 294-297: *function setMaxSupplyExpansionPercents(uint256 _maxSupplyExpansionPercent) external onlyOperator*
- Treasury.sol -> 299-310: *function setSupplyTiersEntry(uint8 _index, uint256 _value) external onlyOperator returns (bool)*
- Treasury.sol -> 312-318: *function setMaxExpansionTiersEntry(uint8 _index, uint256 _value) external onlyOperator returns (bool)*
- Treasury.sol -> 320-323: *function setBondDepletionFloorPercent(uint256 _bondDepletionFloorPercent) external onlyOperator*
- Treasury.sol -> 325-328: *function setMaxSupplyContractionPercent(uint256 _maxSupplyContractionPercent) external onlyOperator*
- Treasury.sol -> 330-333: *function setMaxDebtRatioPercent(uint256 _maxDebtRatioPercent) external onlyOperator*
- Treasury.sol -> 335-340: *function setBootstrap(uint256 _bootstrapEpochs, uint256 _bootstrapSupplyExpansionPercent) external onlyOperator*
- Treasury.sol -> 342-356: *function setExtraFunds(address _daoFund, uint256 _daoFundSharedPercent, address _devFund, uint256 _devFundSharedPercent) external onlyOperator*
- Treasury.sol -> 358-360: *function setMaxDiscountRate(uint256 _maxDiscountRate) external onlyOperator*
- Treasury.sol -> 362-364: *function setMaxPremiumRate(uint256 _maxPremiumRate) external onlyOperator*
- Treasury.sol -> 366-369: *function setDiscountPercent(uint256 _discountPercent) external onlyOperator*
- Treasury.sol -> 371-375: *function setPremiumThreshold(uint256 _premiumThreshold) external onlyOperator*
- Treasury.sol -> 377-380: *function setPremiumPercent(uint256 _premiumPercent) external onlyOperator*

	• Treasury.sol -> 382-385: *function setMintingFactorForPayingDebt(uint256 _mintingFactorForPayingDebt) external onlyOperator*

DESCRIPTION	Functions that change important variables should emit events such that users can more easily monitor the change.
RECOMMENDATION	Emit events from these functions.
RESOLUTION	No changes were made. Reviewed in commit a5c2004bbdae37c3c0533994d9ec2dee558e34f8

On-Chain Analysis

Operator Is An EOA

FINDING ID	#0015
SEVERITY	Low Risk
STATUS	Open
LOCATION	operator

DESCRIPTION	The Operator of GenesisRewardPool.sol and PaeRewardPool.sol is an EOA and the functions that only the operator can call are: • governanceRecoverUnsupported() • setOperator() • add() • set() The Operator of Oracle.sol is an EOA and the function that only the operator can call are: • setPeriod() • setEpoch() The Operator of Treasury.sol is an EOA and the function that only the operator can call are: • masonryGovernanceRecoverUnsupported() • setMaxSupplyExpansionPercents() • setMaxSupplyContractionPercent() • governanceRecoverUnsupported() • setMintingFactorForPayingDebt() • setBondDepletionFloorPercent() • masonryAllocateSeigniorage() • setMaxExpansionTiersEntry() • setMaxDebtRatioPercent() • setPremiumThreshold() • masonrySetOperator() • setMaxPremiumRate() • setMaxDiscountRate() • setPremiumPercent() • setSupplyTiersEntry() • setDiscountPercent() • masonrySetLockUp() • setPegPriceCeiling() • setExtraFunds()

	• setPegOracle() • setBootstrap() • setOperator() • setMasonry() Although the operator can change protocol values without delay, there are variable limiters that do not allow the protocol values to be set arbitrarily low or high.
RECOMMENDATION	Transfer the *operator* to a timelock.
RESOLUTION	N/A

TreasuryFund Is An EOA

FINDING ID	#0016
SEVERITY	Low Risk
STATUS	Open
LOCATION	[TreasuryFund]()

DESCRIPTION	The *treasuryFund* of *GenesisRewardPool.sol* is an EOA.
RECOMMENDATION	Transfer the *treasuryFund* to a timelock.
RESOLUTION	N/A

External Addresses

Externally Owned Accounts

Operator

ACCOUNT	0xDa2d96eADAb3671D9DFC6b2901aA85E99F8f0EB3
USAGE	0x8E54Ef5213E74fda5ed3350aa2c247838A75Edb0 *GenesisRewardPool.operator* 0xb5cc0Ed74dde9F26fBfFCe08FF78227F4Fa86029 *PaeRewardPool.operator* 0x09448876068907827ec15F49A8F1a58C70b04d45 *Oracle.operator* 0xEF50641eEBc7255241D11Ba4317Df824E200016F *Treasury.operator*
IMPACT	• receives elevated permissions as owner, operator, or other

TreasuryFund

ACCOUNT	0x02E36D2A8DF62813D7aB3bB0785DAc07421F2d17
USAGE	0x8E54Ef5213E74fda5ed3350aa2c247838A75Edb0 *GenesisRewardPool.treasuryFund*
IMPACT	• receives the DAO fund of the protocol

External Contracts

These contracts are not part of the audit scope.

Deposit Tokens

ADDRESS	WAVAX 0xB31f66AA3C1e785363F0875A1B74E27b85FD66c7 WETH.e 0x49D5c2BdFfac6CE2BFdB6640F4F80f226bc10bAB USDC.e 0xA7D7079b0FEaD91F3e65f86E8915Cb59c1a4C664 PAE 0x9466Ab927611725B9AF76b9F31B2F879Ff14233d mooAvalancheBIFI 0xCeefB07Ad37ff165A0b03DC7C808fD2E2fC77683 JOE 0x6e84a6216eA6dACC71eE8E6b0a5B7322EEbC0fDd
USAGE	0x8E54Ef5213E74fda5ed3350aa2c247838A75Edb0 *GenesisRewardPool.poolInfo* - Variable
IMPACT	• ERC20 Token

Deposit LP Tokens

ADDRESS	Joe LP pAVAX-WAVAX 0x1179E6AF2794fA9d39316951e868772F96230375 Joe LP PAE-WAVAX 0x6139361Ccd4f40abF3d5D22AA3b72A195010F9AB
USAGE	0x8E54Ef5213E74fda5ed3350aa2c247838A75Edb0 *PaeRewardPool.poolInfo* - Variable
IMPACT	• ERC20 Token

Appendix A - Reviewed Documents

Document	**Address**
distribution/GenesisRewardPool.sol	0x8E54Ef5213E74fda5ed3350aa2c247838A75Edb0
distribution/PaeRewardPool.sol	0xb5cc0Ed74dde9F26fBfFCe08FF78227F4Fa86029
interfaces/IBasisAsset.sol	N/A
interfaces/IDistributor.sol	N/A
interfaces/IERC20.sol	N/A
interfaces/IMasonry.sol	N/A
interfaces/IOracle.sol	N/A
interfaces/ITreasury.sol	N/A
interfaces/IUniswapV2Router.sol	N/A
interfaces/IUniswapV2Pair.sol	N/A
lib/Babylonian.sol	N/A
lib/FixedPoint.sol	N/A
lib/UniswapV2OracleLibrary.sol	N/A
owner/Operator.sol	N/A
utils/ContractGuard.sol	N/A
utils/Epoch.sol	N/A
Bond.sol	0x4f1437a43500B7863c614528e6A15b220904010B
Masonry.sol	0xf5e49b0a960459799F1E9b3f313dFA81D2CE553c
Oracle.sol	0x09448876068907827ec15F49A8F1a58C70b04d45
Pae.sol	0x9466Ab927611725B9AF76b9F31B2F879Ff14233d
PegToken.sol	0x6ca558bd3eaB53DA1B25aB97916dd14bf6CFEe4E

Timelock.sol	N/A
Treasury.sol	0xEF50641eEBc7255241D11Ba4317Df824E200016F

Revisions

Revision 1	505e1761ae4ad308a2042c9eba4b2ef8a3f103e6
Revision 2	3a5f5462f83cd77dba35a459ca42c8724a779c7e
Revision 3	1d56ca27a2d1f060cfa88848c1edfc811cc47b40
Revision 4	a5c2004bbdae37c3c0533994d9ec2dee558e34f8

Imported Contracts

OpenZeppelin	3.4.1

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 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 - Audit Procedure

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

The 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.

The Static analysis is a 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.

The on-chain analysis is the audit of the contracts as they are deployed on the blockchain. 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 contracts 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 practices 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