OBELISK

Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 60	2022-05-31	Donut, ByFixter	Audit Final

Audit Notes

Audit Date	2022-04-28 - 2022-05-31
Auditor/Auditors	ByFixter, thing_theory
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	OB565855547

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	Polaris
Description	The very first algorithmic stablecoins ecosystem on Aurora / NEAR.
Website	https://polarisfinance.io/
Contact	https://twitter.com/PolarisFinance_
Contact information	@eYdr1en | polarisfinance.io#0001 on Discord
Token Name(s)	N/A
Token Short	N/A
Contract(s)	See Appendix A
Code Language	Solidity
Chain	Aurora / Near

Audit of Polaris

Obelisk was commissioned by Polaris on the 27th of April 2022 to conduct a comprehensive audit of Polaris' contracts. The following audit was conducted between the 28th of April 2022 and the 31st of May 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.

After going through all the contracted codes, the auditors found multiple issues with varying degrees of severity. In the code itself, there is a high-risk issue with the possibility of a stale Oracle Price which could set the expected token price to 0. We also found 3 medium-risk issues that could under certain circumstances be negative for the protocol and users. The project team worked to solve these issues in any future deployment and the issues is present in the old deployed contracts POLAR, TRIPOLAR, LUNAR contracts, but the team has fixed the issues in their new Ethernal contract. *Please read individual comments on each issue for further understanding.*

Moving to the on-chain analysis, there is no timelock set which means changes can be done instantaneously. Obelisk recommends setting a long enough timelock for anyone to be able to react to possible changes to the protocol contracts.

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 Polaris 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
Stale Oracle Price	#0001	High Risk	Open/Closed*
Unsupported Token Recovery Is Unrestricted After Pools Expire	#0002	Medium Risk	Mitigated/Closed*
Initialize Can Be Called By Anyone	#0003	Medium Risk	Mitigated/Closed*
Reentrancy Risk Using Token With Code Injection	#0004	Medium Risk	Partially Mitigated/Closed*
No Limits For Discount Rate	#0005	Informational	Open
Redundant Check	#0006	Informational	Open/Closed*
Division Before Multiplication	#0007	Informational	Open/Closed*
No Events Emitted For Changes To Protocol Values	#0008	Informational	Open
Missing Zero Checks	#0009	Informational	Open
Unused Libraries	#0010	Informational	Closed
Unused Functions	#0011	Informational	Open
Contract Values Can Be Constant Or Immutable (Gas Optimization)	#0012	Informational	Open/Closed*
Addresses Are Hard Coded	#0013	Informational	Open/Closed*
Oracle May Drift	#0014	Informational	Open

*Open in POLAR, TRIPOLAR, LUNAR contracts, closed in Ethernal

On-Chain Analysis

Finding	ID	Severity	Status
No Timelock	#0015	High Risk	Partially Mitigated
Masonry getpolarPrice Not Working	#0016	Informational	Open

Findings

Code Analysis

Stale Oracle Price

FINDING ID	#0001
SEVERITY	High Risk
STATUS	Open (Closed in Ethernal)
LOCATION	Oracle.sol -> 84-92 LunarOracle.sol -> 84-92 TripolarOracle.sol -> 84-92

```
1     // note this will always return 0 before update has been called
  successfully for the first time.
2     function consult(address _token, uint256 _amountIn) external view
  returns (uint144 amountOut) {
3         if (_token == token0) {
4             amountOut = price0Average.mul(_amountIn).decode144();
5         } else {
6             require(_token == token1, "Oracle: INVALID_TOKEN");
7             amountOut = price1Average.mul(_amountIn).decode144();
8         }
9     }
```

LOCATION	Treasury.sol -> 159-165

```
1     function getpolarPrice() public view returns (uint256 polarPrice) {
2         try IOracle(polarOracle).consult(polar, 1e18) returns (uint144
  price) {
3             return uint256(price/1e6);
4         } catch {
5             revert("Treasury: failed to consult POLAR price from the
  oracle");
6         }
7     }
```

LOCATION Treasury.sol -> 295-297

```
1     function setpolarOracle(address _polarOracle) external onlyOperator
  {
2         polarOracle = _polarOracle;
3     }
```

LOCATION LunarTreasury.sol -> 155-161

```
1     function getlunarPrice() public view returns (uint256 lunarPrice) {
2         try IOracle(lunarOracle).consult(lunar, 1e18) returns (uint144
  price) {
3             return uint256(price);
4         } catch {
5             revert("Treasury: failed to consult LUNAR price from the
  oracle");
6         }
7     }
```

LOCATION LunarTreasury.sol -> 292-294

```
1     function setlunarOracle(address _lunarOracle) external onlyOperator
  {
2         lunarOracle = _lunarOracle;
3     }
```

LOCATION TripolarTreasury.sol -> 155-161

```
1     function getTripolarPrice() public view returns (uint256
  tripolarPrice) {
2         try IOracle(tripolarOracle).consult(tripolar, 1e18) returns
  (uint144 price) {
3             return uint256(price);
4         } catch {
5             revert("Treasury: failed to consult TRIPOLAR price from the
  oracle");
6         }
7     }
```

LOCATION	TripolarTreasury.sol -> 292-294

```
    function setTripolarOracle(address _tripolarOracle) external onlyOperator {
        tripolarOracle = _tripolarOracle;
    }
```

LOCATION	Ethernal.sol -> 44-55

```
    function _getEthernalPrice() internal view returns (uint256 _ethernalPrice) {
        try IOracle(ethernalOracle).consult(address(this), 1e18) returns (uint144 _price) {
            return uint256(_price);
        } catch {
            revert("ETHRNAL: failed to fetch ETHERNAL price from Oracle");
        }
    }

    function setEthernalOracle(address _ethernalOracle) public onlyOperator {
        require(_ethernalOracle != address(0), "oracle address cannot be 0 address");
        ethernalOracle = _ethernalOracle;
    }
```

DESCRIPTION	If the oracle hasn't been updated consulting it will result in a price of 0. This can occur if the oracle is switched out and it wasn't updated before calling *buyBonds()*. Note: The *_getEthernalPrice* internal view function is currently unused.
RECOMMENDATION	Ensure that the oracle is always in a correct state when calling it.
RESOLUTION	**Project team comment:** "If we ever need to switch out oracle we will immediately call after oracle has been switched out. This is already fixed in Ethernal and Orbital contracts and we will keep it that way in all our future contracts." **Obelisk:** "This issue is still present in older contracts, but the project team has solved the issue in the deployed

	Ethernal contracts (Orbital has not been audited).”

Unsupported Token Recovery Is Unrestricted After Pools Expire

FINDING ID	#0002
SEVERITY	Medium Risk
STATUS	Mitigated
LOCATION	In the *governanceRecoverUnsupported()* functions: • PolarGenesisRewardPool.sol -> 272: *if (block.timestamp < poolEndTime + 90 days)* • LunarGenesisRewardPool.sol -> 286: *if (block.timestamp < poolEndTime + 90 days)* • TripolarGenesisRewardPool.sol -> 272: *if (block.timestamp < poolEndTime + 90 days)* • SpolarRewardPool.sol -> 263: *if (block.timestamp < poolEndTime + 90 days)* • PolarRewardPool.sol -> 273: *if (block.timestamp < epochEndTimes[1] + 30 days)*

DESCRIPTION	After the distribution pools have closed, the operator will be able to transfer any remaining tokens (including user deposits) to an arbitrary address. There is a delay of 30 to 90 days before this can happen.
RECOMMENDATION	Remove the conditional such that users' deposit tokens can never be recovered from the pool except by the user.
RESOLUTION	**Project team comment:** "These txs change GenesisRewardPools to 0x0 address so we are not and won't be able to withdraw remaining tokens ever. This is already fixed in Ethernal and Orbital contracts and we will keep it that way in all our future contracts." PolarRewardPool: tx/0xe051afb553b6c2c3e2acde16064c0679d9615dd76a77f7fe5c6f892c47e57aab PolarGenesisRewardPool: tx/0xb306d5b86c334c9ae5649b0df4acd766d72ed0e5b367d14855205b6dbd235592 LunarGenesisRewardPool: tx/0x8bf63a2a0ff76a918aaafdcbe35d8e052b557e0ed2f8ef7d57a81d4fe1ae9bcf TripolarGenesisRewardPool:

	tx/0x6295dd98eae9c3495b3554e937959453d8c1dffc501935106a2252b3fdc5c11f **Obelisk:** “This issue is mitigated in older contracts, and the project team has solved the issue in the deployed Ethernal contracts (Orbital has not been audited).”

Initialize Can Be Called By Anyone

FINDING ID	#0003
SEVERITY	Medium Risk
STATUS	Mitigated (Closed in Ethernal)
LOCATION	Treasury.sol -> 242-284

```
function initialize(
    address _polar,
    address _pbond,
    address _spolar,
    address _polarOracle,
    address _masonry,
    uint256 _startTime
) public notInitialized {
    // ...
}
```

LOCATION	LunarTreasury.sol -> 238-282

```
function initialize(
    address _lunar,
    address _lbond,
    address _spolar,
    address _lunarOracle,
    address _masonry,
    uint256 _startTime,
    address[] memory _excludedFromTotalSupply
) public notInitialized {
    // ...
}
```

LOCATION	TripolarTreasury.sol -> 238-282

```
function initialize(
    address _tripolar,
    address _tribond,
    address _spolar,
    address _tripolarOracle,
    address _masonry,
    uint256 _startTime,
    address[] memory _excludedFromTotalSupply
) public notInitialized {
    // ...
}
```

DESCRIPTION	Anyone can initialize the contract and become the operator if the deployer does not initialize in the same transaction. The check operator no longer confirms that all of the contracts share the same operator. If different addresses initialize different contracts the checkOperator modifier may pass.
RECOMMENDATION	If the Treasury needs to be changed make sure that the initializer is protected.
RESOLUTION	Contracts were initialized on chain. **Project team comment:** “If we need to change the Treasury, LunarTreasury, TripolarTreasury we will make sure that initializer is protected. This is already fixed in Ethernal and Orbital contracts and we will keep it that way in all our future contracts.” **Obelisk:** “This issue is still present in older contracts, but the project team has solved the issue in the deployed Ethernal contracts (Orbital has not been audited).”

Reentrancy Risk Using Token With Code Injection

FINDING ID	#0004
SEVERITY	Medium Risk
STATUS	Partially Mitigated (Closed in Ethernal)
LOCATION	distribution/PolarGenesisRewardPool.sol -> 203-223

```
1     function deposit(uint256 _pid, uint256 _amount) public {
2         // ...
3
4         if (_amount > 0) {
5             pool.token.safeTransferFrom(_sender, address(this),
  _amount);
6             {
7                 user.amount = user.amount.add(_amount);
8             }
9         }
10        user.rewardDebt =
  user.amount.mul(pool.accPolarPerShare).div(1e18);
11        emit Deposit(_sender, _pid, _amount);
12    }
```

LOCATION	distribution/PolarGenesisRewardPool.sol -> 226-243

```
1     function withdraw(uint256 _pid, uint256 _amount) public {
2         // ...
3
4         if (_amount > 0) {
5             user.amount = user.amount.sub(_amount);
6             pool.token.safeTransfer(_sender, _amount);
7         }
8         user.rewardDebt =
  user.amount.mul(pool.accPolarPerShare).div(1e18);
9         emit Withdraw(_sender, _pid, _amount);
10    }
```

LOCATION distribution/LunarGenesisRewardPool.sol -> 210-236

```
    function deposit(uint256 _pid, uint256 _amount) public {
        // ...

        if (_amount > 0) {
            pool.token.safeTransferFrom(_sender, address(this),
_amount);
            if (pool.daoFee > 0) {
                uint256 _fee = _amount.mul(pool.daoFee).div(10000);
                pool.token.safeTransfer(daoAddress, _fee);
                user.amount = user.amount.add(_amount.sub(_fee));
            }
            else {
                user.amount = user.amount.add(_amount);
            }

        }
        user.rewardDebt =
user.amount.mul(pool.accLunarPerShare).div(1e18);
        emit Deposit(_sender, _pid, _amount);
    }
```

LOCATION distribution/LunarGenesisRewardPool.sol -> 239-256

```
    function withdraw(uint256 _pid, uint256 _amount) public {
        // ...

        if (_amount > 0) {
            user.amount = user.amount.sub(_amount);
            pool.token.safeTransfer(_sender, _amount);
        }
        user.rewardDebt =
user.amount.mul(pool.accLunarPerShare).div(1e18);
        emit Withdraw(_sender, _pid, _amount);
    }
```

LOCATION	distribution/TripolarGenesisRewardPool.sol -> 203-222

```
    function deposit(uint256 _pid, uint256 _amount) public {
        // ...

        if (_amount > 0) {
            pool.token.safeTransferFrom(_sender, address(this),
_amount);
            user.amount = user.amount.add(_amount);

        }
        user.rewardDebt =
user.amount.mul(pool.accTripolarPerShare).div(1e18);
        emit Deposit(_sender, _pid, _amount);
    }
```

LOCATION	distribution/TripolarGenesisRewardPool.sol -> 225-242

```
    function withdraw(uint256 _pid, uint256 _amount) public {
        // ...

        if (_amount > 0) {
            user.amount = user.amount.sub(_amount);
            pool.token.safeTransfer(_sender, _amount);
        }
        user.rewardDebt =
user.amount.mul(pool.accTripolarPerShare).div(1e18);
        emit Withdraw(_sender, _pid, _amount);
    }
```

DESCRIPTION	If a token has code injection (such as a *onReceived* function), then a malicious actor may be able to drain the contract of tokens either via the withdraw or deposit function.
RECOMMENDATION	Follow the checks effects interactions pattern. Alternatively, add a reentrancy guard.
RESOLUTION	Not all token contracts were verified. However, most tokens were confirmed to not have code injections. **Project team comment:** “Polar, Lunar, Tripolar don't have code injections. This is already fixed in Ethernal and Orbital GenesisRewardPools and we will keep it that way in all our future GenesisRewardPools.”

Obelisk: “Polar, Lunar or Tripolar does not have code injection. This issue is still present in older contracts, but the project team has solved the issue in the deployed Ethernal contracts (Orbital has not been audited).”

No Limits For Discount Rate

FINDING ID	#0005
SEVERITY	Informational
STATUS	Open
LOCATION	EthernalTreasury.sol -> 364-370 LunarTreasury.sol -> 365-371 Treasury.sol -> 367-373 TripolarTreasury.sol -> 365-371

```
1     function setMaxDiscountRate(uint256 _maxDiscountRate) external
  onlyOperator {
2         maxDiscountRate = _maxDiscountRate;
3     }
4
5     function setMaxPremiumRate(uint256 _maxPremiumRate) external
  onlyOperator {
6         maxPremiumRate = _maxPremiumRate;
7     }
```

DESCRIPTION	The noted setters have no limits and can therefore change the *maxDiscountRate* and *maxPremiumRate* to any value.
RECOMMENDATION	Add a reasonable limit for these values.
RESOLUTION	**Project team comment:** “We don't see any reason to have a limit for these values.”

Redundant Check

FINDING ID	#0006
SEVERITY	Informational
STATUS	Open (Closed in Ethernal)
LOCATION	Treasury.sol -> 309 Treasury.sol -> 322 LunarTreasury.sol -> 307 LunarTreasury.sol -> 320 TripolarTreasury.sol -> 307 TripolarTreasury.sol -> 320

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

DESCRIPTION	Parameter *_index* is a *uint8* which is always greater than or equal to 0. Also, the message of the require statement does not reflect the check.
RECOMMENDATION	Remove the require statement.
RESOLUTION	**Project team comment:** "If we ever need to redeploy we will fix this. This is already fixed in Ethernal and Orbital contracts and we will keep it that way in all our future contracts." **Obelisk:** "This issue is still present in older contracts, but the project team has solved the issue in the deployed Ethernal contracts (Orbital has not been audited)."

Division Before Multiplication

FINDING ID	#0007
SEVERITY	Informational
STATUS	Open (Closed in Ethernal)
LOCATION	distribution/PolarGenesisRewardPool.sol -> 165-166

```
1            uint256 _polarReward =
_generatedReward.mul(pool.allocPoint).div(totalAllocPoint);
2            accPolarPerShare =
accPolarPerShare.add(_polarReward.mul(1e18).div(tokenSupply));
```

LOCATION	distribution/PolarGenesisRewardPool.sol -> 196-197

```
1            uint256 _polarReward =
_generatedReward.mul(pool.allocPoint).div(totalAllocPoint);
2            pool.accPolarPerShare =
pool.accPolarPerShare.add(_polarReward.mul(1e18).div(tokenSupply));
```

LOCATION	distribution/LunarGenesisRewardPool.sol -> 172-173

```
1            uint256 _lunarReward =
_generatedReward.mul(pool.allocPoint).div(totalAllocPoint);
2            accLunarPerShare =
accLunarPerShare.add(_lunarReward.mul(1e18).div(tokenSupply));
```

LOCATION	distribution/LunarGenesisRewardPool.sol -> 203-204

```
1            uint256 _lunarReward =
_generatedReward.mul(pool.allocPoint).div(totalAllocPoint);
2            pool.accLunarPerShare =
pool.accLunarPerShare.add(_lunarReward.mul(1e18).div(tokenSupply));
```

LOCATION	distribution/TripolarGenesisRewardPool.sol -> 165-166

```
1            uint256 _tripolarReward =
_generatedReward.mul(pool.allocPoint).div(totalAllocPoint);
2            accTripolarPerShare =
accTripolarPerShare.add(_tripolarReward.mul(1e18).div(tokenSupply));
```

LOCATION	distribution/TripolarGenesisRewardPool.sol -> 196-197

```
1            uint256 _tripolarReward =
_generatedReward.mul(pool.allocPoint).div(totalAllocPoint);
2            pool.accTripolarPerShare =
pool.accTripolarPerShare.add(_tripolarReward.mul(1e18).div(tokenSupply)
);
```

DESCRIPTION	The calculations noted use mixed orders of multiplication and division. This may cause rounding errors, resulting in miscalculations.
RECOMMENDATION	Change the calculations to first multiply, then divide. (Make sure there is no way of overflowing)
RESOLUTION	**Project team comment:** “This is fixed in Ethernal and Orbital contracts and we will keep it that way in all our future contracts.” **Obelisk:** “This issue is still present in older contracts, but the project team has solved the issue in the deployed Ethernal contracts (Orbital has not been audited).”

No Events Emitted For Changes To Protocol Values

FINDING ID	#0008
SEVERITY	Informational
STATUS	Open
LOCATION	• distribution/EthernalGenesisRewardPool.sol -> 92-133: *function add(uint256 _allocPoint, IERC20 _token, bool _withUpdate, uint256 _lastRewardTime) public onlyOperator {* • distribution/EthernalGenesisRewardPool.sol -> 136-145: *function set(uint256 _pid, uint256 _allocPoint) public onlyOperator {* • distribution/EthernalGenesisRewardPool.sol -> 276-278: *function setOperator(address _operator) external onlyOperator {* • distribution/LunarGenesisRewardPool.sol -> 95-136: *function add(uint256 _allocPoint, IERC20 _token, bool _withUpdate, uint256 _lastRewardTime) public onlyOperator {* • distribution/LunarGenesisRewardPool.sol -> 139-148: *function set(uint256 _pid, uint256 _allocPoint) public onlyOperator {* • distribution/LunarGenesisRewardPool.sol -> 281-283: *function setOperator(address _operator) external onlyOperator {* • distribution/LunarGenesisRewardPool.sol -> 298-302: *function setDaoFee(uint256 _pid, uint _fee) external onlyOperator {* • distribution/PolarGenesisRewardPool.sol -> 91-129: *function add(uint256 _allocPoint, IERC20 _token, bool _withUpdate, uint256 _lastRewardTime) public onlyOperator {* • distribution/PolarGenesisRewardPool.sol -> 132-141: *function set(uint256 _pid, uint256 _allocPoint) public onlyOperator {* • distribution/PolarGenesisRewardPool.sol -> 268-270: *function setOperator(address _operator) external onlyOperator {* • distribution/PolarRewardPool.sol -> 89-119: *function add(uint256 _allocPoint, IERC20 _token, bool _withUpdate, uint256 _lastRewardTime) public onlyOperator {* • distribution/PolarRewardPool.sol -> 122-129: *function set(uint256 _pid, uint256 _allocPoint) public onlyOperator {* • distribution/PolarRewardPool.sol -> 264-266: *function setOperator(address _operator) external onlyOperator {* • distribution/SpolarRewardPool.sol -> 83-121: *function add(uint256 _allocPoint, IERC20 _token, bool _withUpdate, uint256 _lastRewardTime) public onlyOperator {*

- distribution/SpolarRewardPool.sol -> 124-133: *function set(uint256 _pid, uint256 _allocPoint) public onlyOperator {*
- distribution/SpolarRewardPool.sol -> 258-260: *function setOperator(address _operator) external onlyOperator {*
- distribution/TripolarGenesisRewardPool.sol -> 91-129: *function add(uint256 _allocPoint, IERC20 _token, bool _withUpdate, uint256 _lastRewardTime) public onlyOperator {*
- distribution/TripolarGenesisRewardPool.sol -> 132-141: *function set(uint256 _pid, uint256 _allocPoint) public onlyOperator {*
- distribution/TripolarGenesisRewardPool.sol -> 267-269: *function setOperator(address _operator) external onlyOperator {*
- Ethernal.sol -> 52-55: *function setEthernalOracle(address _ethernalOracle) public onlyOperator {*
- Ethernal.sol -> 104-110: *function governanceRecoverUnsupported(IERC20 _token, uint256 _amount, address _to) external onlyOperator {*
- EthernalSunrise.sol -> 146-148: *function setOperator(address _operator) external onlyOperator {*
- EthernalSunrise.sol -> 150-154: *function setLockUp(uint256 _withdrawLockupEpochs, uint256 _rewardLockupEpochs) external onlyOperator {*
- EthernalSunrise.sol -> 260-265: *function governanceRecoverUnsupported(IERC20 _token, uint256 _amount, address _to) external onlyOperator {*
- EthernalTreasury.sol -> 284-286: *function setOperator(address _operator) external onlyOperator {*
- EthernalTreasury.sol -> 288-290: *function setMasonry(address _masonry) external onlyOperator {*
- EthernalTreasury.sol -> 292-295: *function setEthernalOracle(address _ethernalOracle) external onlyOperator {*
- EthernalTreasury.sol -> 297-300: *function setEthernalPriceCeiling(uint256 _ethernalPriceCeiling) external onlyOperator {*
- EthernalTreasury.sol -> 302-305: *function setMaxSupplyExpansionPercents(uint256 _maxSupplyExpansionPercent) external onlyOperator {*
- EthernalTreasury.sol -> 307-317: *function setSupplyTiersEntry(uint8 _index, uint256 _value) external onlyOperator returns (bool) {*

- EthernalTreasury.sol -> 319-322: *function setMaxExpansionTiersEntry(uint8 _index, uint256 _value) external onlyOperator returns (bool) {*
- EthernalTreasury.sol -> 326-329: *function setbondDepletionFloorPercent(uint256 _bondDepletionFloorPercent) external onlyOperator {*
- EthernalTreasury.sol -> 331-334: *function setMaxSupplyContractionPercent(uint256 _maxSupplyContractionPercent) external onlyOperator {*
- EthernalTreasury.sol -> 336-339: *function setMaxDebtRatioPercent(uint256 _maxDebtRatioPercent) external onlyOperator {*
- EthernalTreasury.sol -> 341-346: *function setBootstrap(uint256 _bootstrapEpochs, uint256 _bootstrapSupplyExpansionPercent) external onlyOperator {*
- EthernalTreasury.sol -> 348-362: *function setExtraFunds(address _daoFund, uint256 _daoFundSharedPercent, address _devFund, uint256 _devFundSharedPercent) external onlyOperator {*
- EthernalTreasury.sol -> 364-366: *function setMaxDiscountRate(uint256 _maxDiscountRate) external onlyOperator {*
- EthernalTreasury.sol -> 368-370: *function setMaxPremiumRate(uint256 _maxPremiumRate) external onlyOperator {*
- EthernalTreasury.sol -> 372-375: *function setDiscountPercent(uint256 _discountPercent) external onlyOperator {*
- EthernalTreasury.sol -> 377-381: *function setPremiumThreshold(uint256 _premiumThreshold) external onlyOperator {*
- EthernalTreasury.sol -> 383-386: *function setPremiumPercent(uint256 _premiumPercent) external onlyOperator {*
- EthernalTreasury.sol -> 388-391: *function setMintingFactorForPayingDebt(uint256 _mintingFactorForPayingDebt) external onlyOperator {*
- EthernalTreasury.sol -> 541-551: *function governanceRecoverUnsupported(IERC20 _token, uint256 _amount, address _to) external onlyOperator*

- Lunar.sol -> 55-58: *function setLunarOracle(address _lunarOracle) public onlyOperator {*
- Lunar.sol -> 116-122: *function governanceRecoverUnsupported(IERC20 _token, uint256 _amount, address _to) external onlyOperator {*
- LunarSunrise.sol -> 146-148: *function setOperator(address _operator) external onlyOperator {*
- LunarSunrise.sol -> 150-154: *function setLockUp(uint256 _withdrawLockupEpochs, uint256 _rewardLockupEpochs) external onlyOperator {*
- LunarSunrise.sol -> 260-265: *function governanceRecoverUnsupported(IERC20 _token, uint256 _amount, address _to) external onlyOperator {*
- LunarTreasury.sol -> 284-286: *function setOperator(address _operator) external onlyOperator {*
- LunarTreasury.sol -> 288-290: *function setMasonry(address _masonry) external onlyOperator {*
- LunarTreasury.sol -> 292-294: *function setlunarOracle(address _lunarOracle) external onlyOperator {*
- LunarTreasury.sol -> 296-299: *function setlunarPriceCeiling(uint256 _lunarPriceCeiling) external onlyOperator {*
- LunarTreasury.sol -> 301-304: *function setMaxSupplyExpansionPercents(uint256 _maxSupplyExpansionPercent) external onlyOperator {*
- LunarTreasury.sol -> 306-317: *function setSupplyTiersEntry(uint8 _index, uint256 _value) external onlyOperator returns (bool) {*
- LunarTreasury.sol -> 319-325: *function setMaxExpansionTiersEntry(uint8 _index, uint256 _value) external onlyOperator returns (bool) {*
- LunarTreasury.sol -> 327-330: *function sepBondDepletionFloorPercent(uint256 _bondDepletionFloorPercent) external onlyOperator {*
- LunarTreasury.sol -> 332-335: *function setMaxSupplyContractionPercent(uint256 _maxSupplyContractionPercent) external onlyOperator {*
- LunarTreasury.sol -> 337-340: *function setMaxDebtRatioPercent(uint256 _maxDebtRatioPercent) external onlyOperator {*

- LunarTreasury.sol -> 342-347: *function setBootstrap(uint256 _bootstrapEpochs, uint256 _bootstrapSupplyExpansionPercent) external onlyOperator {*
- LunarTreasury.sol -> 349-363: *function setExtraFunds(address _daoFund, uint256 _daoFundSharedPercent, address _devFund, uint256 _devFundSharedPercent) external onlyOperator {*
- LunarTreasury.sol -> 365-367: *function setMaxDiscountRate(uint256 _maxDiscountRate) external onlyOperator {*
- LunarTreasury.sol -> 369-371: *function setMaxPremiumRate(uint256 _maxPremiumRate) external onlyOperator {*
- LunarTreasury.sol -> 373-376: *function setDiscountPercent(uint256 _discountPercent) external onlyOperator {*
- LunarTreasury.sol -> 378-382: *function setPremiumThreshold(uint256 _premiumThreshold) external onlyOperator {*
- LunarTreasury.sol -> 384-387: *function setPremiumPercent(uint256 _premiumPercent) external onlyOperator {*
- LunarTreasury.sol -> 389-392: *function setMintingFactorForPayingDebt(uint256 _mintingFactorForPayingDebt) external onlyOperator {*
- LunarTreasury.sol -> 542-552: *function governanceRecoverUnsupported(IERC20 _token, uint256 _amount, address _to) external onlyOperator*
- Masonry.sol -> 146-148: *function setOperator(address _operator) external onlyOperator {*
- Masonry.sol -> 150-154: *function setLockUp(uint256 _withdrawLockupEpochs, uint256 _rewardLockupEpochs) external onlyOperator {*
- Masonry.sol -> 260-265: *function governanceRecoverUnsupported(IERC20 _token, uint256 _amount, address _to) external onlyOperator {*
- Polar.sol -> 54-57: *function setPolarOracle(address _polarOracle) public onlyOperator {*
- Polar.sol -> 109-115: *function governanceRecoverUnsupported(IERC20 _token, uint256 _amount, address _to) external onlyOperator {*

- Spolar.sol -> 62-65: *function setTreasuryFund(address _communityFund) external {*
- Spolar.sol -> 67-71: *function setDevFund(address _devFund) external {*
- Spolar.sol -> 117-123: *function governanceRecoverUnsupported(IERC20 _token, uint256 _amount, address _to) external onlyOperator {*
- Treasury.sol -> 286-288: *function setOperator(address _operator) external onlyOperator {*
- Treasury.sol -> 290-292: *function setMasonry(address _masonry) external onlyOperator {*
- Treasury.sol -> 294-296: *function setpolarOracle(address _polarOracle) external onlyOperator {*
- Treasury.sol -> 298-301: *function setpolarPriceCeiling(uint256 _polarPriceCeiling) external onlyOperator {*
- Treasury.sol -> 303-306: *function setMaxSupplyExpansionPercents(uint256 _maxSupplyExpansionPercent) external onlyOperator {*
- Treasury.sol -> 308-319: *function setSupplyTiersEntry(uint8 _index, uint256 _value) external onlyOperator returns (bool) {*
- Treasury.sol -> 321-327: *function setMaxExpansionTiersEntry(uint8 _index, uint256 _value) external onlyOperator returns (bool) {*
- Treasury.sol -> 329-332: *function sepBondDepletionFloorPercent(uint256 _bondDepletionFloorPercent) external onlyOperator {*
- Treasury.sol -> 334-337: *function setMaxSupplyContractionPercent(uint256 _maxSupplyContractionPercent) external onlyOperator {*
- Treasury.sol -> 339-342: *function setMaxDebtRatioPercent(uint256 _maxDebtRatioPercent) external onlyOperator {*
- Treasury.sol -> 344-349: *function setBootstrap(uint256 _bootstrapEpochs, uint256 _bootstrapSupplyExpansionPercent) external onlyOperator {*
- Treasury.sol -> 351-365: *function setExtraFunds(address _daoFund, uint256 _daoFundSharedPercent, address _devFund, uint256 _devFundSharedPercent) external onlyOperator {*
- Treasury.sol -> 367-369: *function setMaxDiscountRate(uint256 _maxDiscountRate) external onlyOperator {*

- Treasury.sol -> 371-373: *function setMaxPremiumRate(uint256 _maxPremiumRate) external onlyOperator {*
- Treasury.sol -> 375-378: *function setDiscountPercent(uint256 _discountPercent) external onlyOperator {*
- Treasury.sol -> 380-384: *function setPremiumThreshold(uint256 _premiumThreshold) external onlyOperator {*
- Treasury.sol -> 386-389: *function setPremiumPercent(uint256 _premiumPercent) external onlyOperator {*
- Treasury.sol -> 391-394: *function setMintingFactorForPayingDebt(uint256 _mintingFactorForPayingDebt) external onlyOperator {*
- Treasury.sol -> 544-554: *function governanceRecoverUnsupported(IERC20 _token, uint256 _amount, address _to) external onlyOperator*
- Tripolar.sol -> 52-55: *function setTripolarOracle(address _tripolarOracle) public onlyOperator {*
- Tripolar.sol -> 104-110: *function governanceRecoverUnsupported(IERC20 _token, uint256 _amount, address _to) external onlyOperator {*
- TripolarSunrise.sol -> 146-148: *function setOperator(address _operator) external onlyOperator {*
- TripolarSunrise.sol -> 150-154: *function setLockUp(uint256 _withdrawLockupEpochs, uint256 _rewardLockupEpochs) external onlyOperator {*
- TripolarSunrise.sol -> 260-265: *function governanceRecoverUnsupported(IERC20 _token, uint256 _amount, address _to) external onlyOperator {*
- TripolarTreasury.sol -> 284-286: *function setOperator(address _operator) external onlyOperator {*
- TripolarTreasury.sol -> 288-290: *function setMasonry(address _masonry) external onlyOperator {*
- TripolarTreasury.sol -> 292-294: *function settripolarOracle(address _tripolarOracle) external onlyOperator {*
- TripolarTreasury.sol -> 296-299: *function settripolarPriceCeiling(uint256 _tripolarPriceCeiling) external onlyOperator {*
- TripolarTreasury.sol -> 301-304: *function setMaxSupplyExpansionPercents(uint256 _maxSupplyExpansionPercent) external onlyOperator {*

- TripolarTreasury.sol -> 306-317: *function setSupplyTiersEntry(uint8 _index, uint256 _value) external onlyOperator returns (bool) {*
- TripolarTreasury.sol -> 319-325: *function setMaxExpansionTiersEntry(uint8 _index, uint256 _value) external onlyOperator returns (bool) {*
- TripolarTreasury.sol -> 327-330: *function sepBondDepletionFloorPercent(uint256 _bondDepletionFloorPercent) external onlyOperator {*
- TripolarTreasury.sol -> 332-335: *function setMaxSupplyContractionPercent(uint256 _maxSupplyContractionPercent) external onlyOperator {*
- TripolarTreasury.sol -> 337-340: *function setMaxDebtRatioPercent(uint256 _maxDebtRatioPercent) external onlyOperator {*
- TripolarTreasury.sol -> 342-347: *function setBootstrap(uint256 _bootstrapEpochs, uint256 _bootstrapSupplyExpansionPercent) external onlyOperator {*
- TripolarTreasury.sol -> 349-363: *function setExtraFunds(address _daoFund, uint256 _daoFundSharedPercent, address _devFund, uint256 _devFundSharedPercent) external onlyOperator {*
- TripolarTreasury.sol -> 365-367: *function setMaxDiscountRate(uint256 _maxDiscountRate) external onlyOperator {*
- TripolarTreasury.sol -> 369-371: *function setMaxPremiumRate(uint256 _maxPremiumRate) external onlyOperator {*
- TripolarTreasury.sol -> 373-376: *function setPremiumThreshold(uint256 _discountPercent) external onlyOperator {*
- TripolarTreasury.sol -> 378-382: *function setDiscountPercent(uint256 _premiumThreshold) external onlyOperator {*
- TripolarTreasury.sol -> 384-387: *function setPremiumPercent(uint256 _premiumPercent) external onlyOperator {*
- TripolarTreasury.sol -> 389-392: *function setMintingFactorForPayingDebt(uint256 _mintingFactorForPayingDebt) external onlyOperator {*

	• TripolarTreasury.sol -> 542-552: *function governanceRecoverUnsupported(IERC20 _token, uint256 _amount, address _to) 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	N/A

Missing Zero Checks

FINDING ID	#0009
SEVERITY	Informational
STATUS	Open
LOCATION	• distribution/EthernalGenesisRewardPool.sol -> 92-133: *function add(uint256 _allocPoint, IERC20 _token, bool _withUpdate, uint256 _lastRewardTime, uint _daoFee) public onlyOperator {* • distribution/EthernalGenesisRewardPool.sol -> 276-278: *function setOperator(address _operator) external onlyOperator {* • distribution/LunarGenesisRewardPool.sol -> 95-136: *function add(uint256 _allocPoint, IERC20 _token, bool _withUpdate, uint256 _lastRewardTime) public onlyOperator {* • distribution/LunarGenesisRewardPool.sol -> 281-283: *function setOperator(address _operator) external onlyOperator {* • distribution/PolarGenesisRewardPool.sol -> 91-129: *function add(uint256 _allocPoint, IERC20 _token, bool _withUpdate, uint256 _lastRewardTime) public onlyOperator {* • distribution/PolarGenesisRewardPool.sol -> 268-270: *function setOperator(address _operator) external onlyOperator {* • distribution/PolarRewardPool.sol -> 89-119: *function add(uint256 _allocPoint, IERC20 _token, bool _withUpdate, uint256 _lastRewardTime) public onlyOperator {* • distribution/PolarRewardPool.sol -> 264-266: *function setOperator(address _operator) external onlyOperator {* • distribution/SpolarRewardPool.sol -> 83-121: *function add(uint256 _allocPoint, IERC20 _token, bool _withUpdate, uint256 _lastRewardTime) public onlyOperator {* • distribution/TripolarGenesisRewardPool.sol -> 91-129: *function add(uint256 _allocPoint, IERC20 _token, bool _withUpdate, uint256 _lastRewardTime) public onlyOperator {* • distribution/TripolarGenesisRewardPool.sol -> 267-269: *function setOperator(address _operator) external onlyOperator {* • distribution/SpolarRewardPool.sol -> 258-260: *function setOperator(address _operator) external onlyOperator {* • Ethernal.sol -> 52-55: *function setEthernalOracle(address _ethernalOracle) public onlyOperator {*

- EthernalSunrise.sol -> 146-148: *function setOperator(address _operator) external onlyOperator {*
- EthernalTreasury.sol -> 284-286: *function setOperator(address _operator) external onlyOperator {*
- EthernalTreasury.sol -> 288-290: *function setMasonry(address _masonry) external onlyOperator {*
- EthernalTreasury.sol -> 292-295: *function setEthernalOracle(address _ethernalOracle) external onlyOperator {*
- EthernalTreasury.sol -> 348-362: *function setExtraFunds(address _daoFund, uint256 _daoFundSharedPercent, address _devFund, uint256 _devFundSharedPercent) external onlyOperator {*
- LunarSunrise.sol -> 149-151: *function setOperator(address _operator) external onlyOperator {*
- LunarTreasury.sol -> 284-286: *function setOperator(address _operator) external onlyOperator {*
- LunarTreasury.sol -> 288-290: *function setMasonry(address _masonry) external onlyOperator {*
- LunarTreasury.sol -> 292-294: *function setlunarOracle(address _lunarOracle) external onlyOperator {*
- LunarTreasury.sol -> 349-363: *function setExtraFunds(address _daoFund, uint256 _daoFundSharedPercent, address _devFund, uint256 _devFundSharedPercent) external onlyOperator {*
- Masonry.sol -> 146-148: *function setOperator(address _operator) external onlyOperator {*
- Spolar.sol -> 62-65: *function setTreasuryFund(address _communityFund) external {*
- Treasury.sol -> 286-288: *function setOperator(address _operator) external onlyOperator {*
- Treasury.sol -> 290-292: *function setMasonry(address _masonry) external onlyOperator {*
- Treasury.sol -> 294-296: *function setpolarOracle(address _polarOracle) external onlyOperator {*
- Treasury.sol -> 351-365: *function setExtraFunds(address _daoFund, uint256 _daoFundSharedPercent, address _devFund, uint256 _devFundSharedPercent) external onlyOperator {*
- TripolarSunrise.sol -> 148-150: *function setOperator(address _operator) external onlyOperator {*
- TripolarTreasury.sol -> 284-286: *function setOperator(address _operator) external onlyOperator {*

	• TripolarTreasury.sol -> 288-290: *function setMasonry(address _masonry) external onlyOperator {* • TripolarTreasury.sol -> 292-294: *function settripolarOracle(address _tripolarOracle) external onlyOperator {* • TripolarTreasury.sol -> 349-363: *function setExtraFunds(address _daoFund, uint256 _daoFundSharedPercent, address _devFund, uint256 _devFundSharedPercent) 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	N/A

Unused Libraries

FINDING ID	#0010
SEVERITY	Informational
STATUS	Closed
LOCATION	• libs/Safe112.sol • libs/UQ112x112.sol

DESCRIPTION	The noted libraries are never used.
RECOMMENDATION	Remove the files.
RESOLUTION	The files have been removed.

Unused Functions

FINDING ID	#0011
SEVERITY	Informational
STATUS	Open
LOCATION	• lib/Babylonian.sol -> 4-16: *Babylonian.sqrt(uint256)* • lib/FixedPoint.sol -> 55-57: *FixedPoint.decode(FixedPoint.uq112x112)* • lib/FixedPoint.sol -> 34-37: *FixedPoint.div(FixedPoint.uq112x112,uint112)* • lib/FixedPoint.sol -> 24-26: *FixedPoint.encode(uint112)* • lib/FixedPoint.sol -> 29-31: *FixedPoint.encode144(uint144)* • lib/FixedPoint.sol -> 65-68: *FixedPoint.reciprocal(FixedPoint.uq112x112)* • lib/FixedPoint.sol -> 71-73: *FixedPoint.sqrt(FixedPoint.uq112x112)* • lib/SafeMath8.sol -> 29-34: *SafeMath8.add(uint8,uint8)* • lib/SafeMath8.sol -> 103-105: *SafeMath8.div(uint8,uint8)* • lib/SafeMath8.sol -> 119-125: *SafeMath8.div(uint8,uint8,string)* • lib/SafeMath8.sol -> 139-141: *SafeMath8.mod(uint8,uint8)* • lib/SafeMath8.sol -> 155-158: *SafeMath8.mod(uint8,uint8,string)* • lib/SafeMath8.sol -> 77-89: *SafeMath8.mul(uint8,uint8)* • lib/UniswapV2Library -> 104-117: *UniswapV2Library.getAmountIn(uint256,uint256,uint256)* • lib/UniswapV2Library -> 89-101: *UniswapV2Library.getAmountOut(uint256,uint256,uint256)* • lib/UniswapV2Library -> 76-86: *UniswapV2Library.getAmountsIn(address,uint256,address[])* • lib/UniswapV2Library -> 62-73: *UniswapV2Library.getAmountsOut(address,uint256,address[])* • lib/UniswapV2Library -> 40-48: *UniswapV2Library.getReserves(address,address,address)* • lib/UniswapV2Library -> 19-37: *UniswapV2Library.pairFor(address,address,address)* • lib/UniswapV2Library -> 51-59: *UniswapV2Library.quote(uint256,uint256,uint256)* • lib/UniswapV2Library -> 12-16: *UniswapV2Library.sortTokens(address,address)*

DESCRIPTION	The noted functions are never used.
RECOMMENDATION	Remove the functions.
RESOLUTION	N/A

Contract Values Can Be Constant Or Immutable (Gas Optimization)

FINDING ID	#0012
SEVERITY	Informational
STATUS	Open (Closed in Ethernal)
LOCATION	• distribution/PolarGenesisRewardPool.sol -> 57: *uint256 public polarPerSecond = 0.0462962963e18;* • distribution/PolarGenesisRewardPool.sol -> 58: *uint256 public runningTime = 1 days;* • distribution/LunarGenesisRewardPool.sol -> 59: *uint256 public lunarPerSecond = 0.0081018519e18;* • distribution/LunarGenesisRewardPool.sol -> 60: *uint256 public runningTime = 1 days;* • distribution/SpolarRewardPool.sol -> 50: *uint256 public spolarPerSecond = 0.0015817901e18;* • distribution/SpolarRewardPool.sol -> 51: *uint256 public runningTime = 300 days;* • distribution/TripolarGenesisRewardPool.sol -> 59: *uint256 public tripolarPerSecond = 0.5787037037e18;* • distribution/TripolarGenesisRewardPool.sol -> 60: *uint256 public runningTime = 1 days;*

DESCRIPTION	Variables which do not change during the operation of a contract can be marked *constant* or *immutable* to reduce gas costs and improve code readability.
RECOMMENDATION	Mark these variables as *constant* or *immutable* as appropriate.
RESOLUTION	**Project team comment:** “As aurora is really cheap this is not an issue to anyone. However this is already fixed in Ethernal and Orbital contracts and we will keep it that way in all our future contracts.” **Obelisk:** “This issue is still present in older contracts, but the project team has solved the issue in the deployed Ethernal contracts (Orbital has not been audited).”

Addresses Are Hard Coded

FINDING ID	#0013
SEVERITY	Informational
STATUS	Open
LOCATION	Treasury.sol 49-52

```
1     address[] public excludedFromTotalSupply = [
2         address(0x7afd06811120462e86dbd41554445928C7AFBCAD), //
  polarGenesisPool
3         address(0x5749c1374cc9DE28B94cf054dEb303D4cA3464bF) // new
  polarRewardPool
4     ];
```

LOCATION	Tripolar.sol 40-41

```
1         // Mints 50000 TRIPOLAR to DAO for initial pool setup
2         _mint(0xFD7cbEAe92ED5ec9C58b88FE5aF9A78115A82cd8, 50000e18);
```

DESCRIPTION	The noted token addresses are hard coded.
RECOMMENDATION	Add parameters to the constructor to allow for more flexible deployment.
RESOLUTION	**Project team comment:** “We will have this in mind for our future contracts. This doesn't affect users at all.” **Obelisk:** “This issue is still present in older contracts, but the project team has solved the issue in the deployed Ethernal contracts (Orbital has not been audited).”

Oracle May Drift

FINDING ID	#0014
SEVERITY	Informational
STATUS	Open
LOCATION	EthernalOracle.sol -> 63-82 PolarOracle.sol -> 63-82 TripolarOracle.sol -> 63-82 Oracle.sol -> 63-82

```
1     function update() external checkEpoch {
2         (uint256 price0Cumulative, uint256 price1Cumulative, uint32
  blockTimestamp) =
  UniswapV2OracleLibrary.currentCumulativePrices(address(pair));
3         uint32 timeElapsed = blockTimestamp - blockTimestampLast; //
  overflow is desired
4
5         if (timeElapsed == 0) {
6             // prevent divided by zero
7             return;
8         }
9
10        // overflow is desired, casting never truncates
11        // cumulative price is in (uq112x112 price * seconds) units so
  we simply wrap it after division by time elapsed
12        price0Average = FixedPoint.uq112x112(uint224((price0Cumulative
  - price0CumulativeLast) / timeElapsed));
13        price1Average = FixedPoint.uq112x112(uint224((price1Cumulative
  - price1CumulativeLast) / timeElapsed));
14
15        price0CumulativeLast = price0Cumulative;
16        price1CumulativeLast = price1Cumulative;
17        blockTimestampLast = blockTimestamp;
18
19        emit Updated(price0Cumulative, price1Cumulative);
20    }
```

DESCRIPTION	The *checkEpoch* modifier ensures that the contract is only updated within a certain epoch. If the oracle is updated after the epoch starts, then the Oracle *PERIOD* may cause the update to occur later and later relative to the epoch start.
RECOMMENDATION	Ensure that this is intended behavior as the Oracle update could occur near the end of an epoch or could miss an epoch entirely.

RESOLUTION	N/A

On-Chain Analysis

No Timelock

FINDING ID	#0015
SEVERITY	High Risk
STATUS	Partially Mitigated
LOCATION	(see below)

DESCRIPTION	The *operator* role of the following contracts has not been transferred to a timelock: EthernalGenesisRewardPool 0xc3d490F6369e709f6FA79DA673Cc74F21764ecF7 EthernalOracle 0x2885a5b030bB979f7F19c692b3C9F1C7Bc8829E3 EthernalTreasury 0x8a7FFE979095F0ee1Fed23B455d25b46d11B6B9D PolarGenesisRewardPool 0x7afd06811120462e86dbd41554445928C7AFBCAD PolarOracle 0x73E2bBcb511C7De83f489205788304c3Bc31A119 PolarRewardPool 0x5749c1374cc9DE28B94cf054dEb303D4cA3464bF SpolarRewardPool 0xA5dF6D8D59A7fBDb8a11E23FDa9d11c4103dc49f Treasury 0x2Af7e5b19405A02FC99468Af38a23aa270921781 Tribond 0x8200B4F47eDb608e36561495099a8caF3F806198 Tripolar 0x60527a2751A827ec0Adf861EfcAcbf111587d748 TripolarGenesisRewardPool 0xa003bFBE51CeB23a57C82128bC0c5B2422B63CE1 TripolarOracle 0xad054C533ea23486eEc6060975eb8B4Ad35fc4E2 TripolarSunrise 0x203a65b3153C55B57f911Ea73549ed0b8EC82B2D TripolarTreasury 0x116449c7B1d0837439E12216d7b1EbaecbD848FF The *owner* role of the following contracts has not been transferred to a timelock:

	Ethernal 0x17cbd9C274e90C537790C51b4015a65cD015497e Ebond 0x266437E6c7500A947012F19A3dE96a3881a0449E EthernalOracle 0x2885a5b030bB979f7F19c692b3C9F1C7Bc8829E3 Pbond 0x3a4773e600086A753862621A26a2E3274610da43 PolarOracle 0x73E2bBcb511C7De83f489205788304c3Bc31A119 Spolar 0x9D6fc90b25976E40adaD5A3EdD08af9ed7a21729 Tribond 0x8200B4F47eDb608e36561495099a8caF3F806198 Tripolar 0x60527a2751A827ec0Adf861EfcAcbf111587d748 TripolarOracle 0xad054C533ea23486eEc6060975eb8B4Ad35fc4E2
RECOMMENDATION	Transfer the *operator* and *owner* roles to a timelock contract. Obelisk recommends a minimum delay of 72 hrs.
RESOLUTION	**Project team comment:** Treasury -> We need to be owner for this contract because we sometimes need to make fast changes to expansion rates because of the fast moving nature of crypto and the crypto prices. PolarGenesisRewardPool -> Transfered to 0x0000000000000000000000000000000000000000 PolarRewardPool -> Transfered to 0x0000000000000000000000000000000000000000 PolarOracle -> Operator is transfered to Treasury. Also we are not and won't be able to manipulate anything even with Operator/Owner. Spolar -> This doesn't make any sense. We can't mint or literally do anything even as an Operator/Owner. SpolarRewardPool -> We have a bot running that is balancing pools distribution of spolar between each other after each and every epoch so this is not possible to use. Also with our expansion of our project we will be adding new LPs (pools) to this contracts.. Again this is not possible. TripolarSunrise -> Operator is set to TripolarTreasury. Tribond -> We need to be owner for this contract because we sometimes need to make fast changes to expansion rates because of the fast moving nature of crypto and the

crypto prices. Operator is set to TripolarTreasury so if we ever have to replace TripolarTreasury we will be able to do so.
Tripolar -> We need to be owner for this contract because we sometimes need to make fast changes to expansion rates because of the fast moving nature of crypto and the crypto prices. Operator is set to TripolarTreasury so if we ever have to replace TripolarTreasury we will be able to do so.
TripolarTreasury -> We need to be owner for this contract because we sometimes need to make fast changes to expansion rates because of the fast moving nature of crypto and the crypto prices.
TripolarGenesisRewardPool -> Transferred to 0x0000000000000000000000000000000000000000
TripolarOracle -> Operator is transferred to Treasury. Also we are not / won't be able to manipulate anything even with Operator/Owner.

Obelisk comment:
The operator/owner is renounced (transferred to 0x0000000000000000000000000000000000000000) for the aforementioned contracts.
The oracles have epochs that can be freely changed by the operator.

Note: The treasury contracts do not have a timelock

Masonry getpolarPrice Not Working

FINDING ID	#0016
SEVERITY	Informational
STATUS	Open
LOCATION	Masonry.sol -> 192-194

```
1    function getPolarPrice() external view returns (uint256) {
2        return treasury.getPolarPrice();
3    }
```

DESCRIPTION	The correct funtion is *getpolarPrice()* with a non capital p.
RECOMMENDATION	Change to the correct function name.
RESOLUTION	N/A

External Addresses

Externally Owned Accounts

Owner/Operator

ACCOUNT	0x3160F7328DF59C14D85DFd09aDdAD4EF18aE3e2c
USAGE	0xc3d490F6369e709f6FA79DA673Cc74F21764ecF7 *EthernalGenesisRewardPool.operator* - Variable 0x6aB21bB15Ef208c37A7c8794EBd9ac422cB755a3 *LunarGenesisRewardPool.operator* - Variable 0x7afd06811120462e86dbd41554445928C7AFBCAD *PolarGenesisRewardPool.operator* - Variable 0x5749c1374cc9DE28B94cf054dEb303D4cA3464bF *PolarRewardPool.polar* - Variable 0xA5dF6D8D59A7fBDb8a11E23FDa9d11c4103dc49f *SpolarRewardPool.operator* - Variable 0xa003bFBE51CeB23a57C82128bC0c5B2422B63CE1 *TripolarGenesisRewardPool.operator* - Variable 0x266437E6c7500A947012F19A3dE96a3881a0449E *Ebond.owner* - Variable 0x17cbd9C274e90C537790C51b4015a65cD015497e *Ethernal.owner* - Variable 0x2885a5b030bB979f7F19c692b3C9F1C7Bc8829E3 *EthernalOracle.operator* - Variable *EthernalOracle.owner* - Variable 0x8a7FFE979095F0ee1Fed23B455d25b46d11B6B9D *EthernalTreasury.devFund* - Variable *EthernalTreasury.operator* - Variable 0x3a101bA3f4a39C921A171473592D4EBDA6bD0B57

Lbond.operator - Variable
Lbond.owner - Variable

0x25e801Eb75859Ba4052C4ac4233ceC0264eaDF8c
Lunar.operator - Variable
Lunar.owner - Variable

0xb028D816B5f679c2aABBEa809EA70D523F8c7707
LunarOracle.operator - Variable
LunarOracle.owner - Variable

0x3A859a07C3b931dcaEC1940c24a4aEfE20D53177
LunarTreasury.devFund - Variable
LunarTreasury.operator - Variable

0x73E2bBcb511C7De83f489205788304c3Bc31A119
Oracle.operator - Variable
Oracle.owner - Variable

0x3a4773e600086A753862621A26a2E3274610da43
Pbond.owner - Variable

0x9D6fc90b25976E40adaD5A3EdD08af9ed7a21729
Spolar.devFund - Variable
Spolar.owner - Variable

0x2Af7e5b19405A02FC99468Af38a23aa270921781
Treasury.devFund - Variable
Treasury.operator - Variable

0x8200B4F47eDb608e36561495099a8caF3F806198
Tribond.operator - Variable

0x60527a2751A827ec0Adf861EfcAcbf111587d748
Tripolar.owner - Variable

0xad054C533ea23486eEc6060975eb8B4Ad35fc4E2
TripolarOracle.operator - Variable
TripolarOracle.owner - Variable

0x116449c7B1d0837439E12216d7b1EbaecbD848FF

	TripolarTreasury.devFund - Variable *TripolarTreasury.operator* - Variable
IMPACT	• receives elevated permissions as owner, operator, or other • receives transfer of tokens deposited or minted by project

External Contracts

These contracts are not part of the audit scope.

DAO/Community Fund

ACCOUNT	0xFD7cbEAe92ED5ec9C58b88FE5aF9A78115A82cd8
USAGE	0xc3d490F6369e709f6FA79DA673Cc74F21764ecF7 *EthernalGenesisRewardPool.daoAddress* - Variable 0x6aB21bB15Ef208c37A7c8794EBd9ac422cB755a3 *LunarGenesisRewardPool.daoAddress* - Variable 0x8a7FFE979095F0ee1Fed23B455d25b46d11B6B9D *EthernalTreasury.daoFund* - Variable 0x3A859a07C3b931dcaEC1940c24a4aEfE20D53177 *LunarTreasury.daoFund* - Variable 0x9D6fc90b25976E40adaD5A3EdD08af9ed7a21729 *Spolar.communityFund* - Variable 0x2Af7e5b19405A02FC99468Af38a23aa270921781 *Treasury.daoFund* - Variable 0x116449c7B1d0837439E12216d7b1EbaecbD848FF *TripolarTreasury.daoFund* - Variable
IMPACT	• receives transfer of tokens deposited or minted by project

External Tokens

These contracts are not part of the audit scope.

Trisolaris LP Token - Ethernal

ADDRESS	0x81D77f8e86f65b9C0F393afe0FC743D888c2d4d7
USAGE	0x2885a5b030bB979f7F19c692b3C9F1C7Bc8829E3 *EthernalOracle.pair* - Variable no setter
IMPACT	• ERC20 Token

Wrapped Ether

ADDRESS	0xC9BdeEd33CD01541e1eeD10f90519d2C06Fe3feB
USAGE	0x2885a5b030bB979f7F19c692b3C9F1C7Bc8829E3 *EthernalOracle.token1* - Variable no setter
IMPACT	• ERC20 Token

Trisolaris LP Token - Lunar

ADDRESS	0x3e50da46cB79d1f9F08445984f207278796CE2d2
USAGE	0xb028D816B5f679c2aABBEa809EA70D523F8c7707 *LunarOracle.pair* - Variable no setter
IMPACT	• ERC20 Token

Luna Terra

ADDRESS	0xC4bdd27c33ec7daa6fcfd8532ddB524Bf4038096
USAGE	0xb028D816B5f679c2aABBEa809EA70D523F8c7707 *LunarOracle.token1* - Variable no setter
IMPACT	• ERC20 Token

Trisolaris LP Token - Polar

ADDRESS	0x3fa4d0145a0b6Ad0584B1ad5f61cB490A04d8242

USAGE	0x73E2bBcb511C7De83f489205788304c3Bc31A119 *Oracle.pair* - Variable no setter
IMPACT	• ERC20 Token

Near Token

ADDRESS	0xC42C30aC6Cc15faC9bD938618BcaA1a1FaE8501d
USAGE	0x73E2bBcb511C7De83f489205788304c3Bc31A119 *Oracle.token0* - Variable no setter
IMPACT	• ERC20 Token

Trisolaris LP Token - Tripolar

ADDRESS	0x85f155FDCf2a951fd95734eCEB99F875b84a2E27
USAGE	0xad054C533ea23486eEc6060975eb8B4Ad35fc4E2 *TripolarOracle.pair* - Variable no setter
IMPACT	• ERC20 Token

TriBar Token

ADDRESS	0x802119e4e253D5C19aA06A5d567C5a41596D6803
USAGE	0xad054C533ea23486eEc6060975eb8B4Ad35fc4E2 *TripolarOracle.token1* - Variable no setter
IMPACT	• ERC20 Token

Appendix A - Reviewed Documents

Deployed Contracts

Document	Address
distribution/EthernalGenesisRewardPool.sol	0xc3d490F6369e709f6FA79DA673Cc74F21764ecF7
distribution/LunarGenesisReward.sol	0x6aB21bB15Ef208c37A7c8794EBd9ac422cB755a3
distribution/PolarGenesisReward.sol	0x7afd06811120462e86dbd41554445928C7AFBCAD
distribution/PolarRewardPool.sol	0x5749c1374cc9DE28B94cf054dEb303D4cA3464bF
distribution/SpolarRewardPool.sol	0xA5dF6D8D59A7fBDb8a11E23FDa9d11c4103dc49f
distribution/TripolarGenesisReward.sol	0xa003bFBE51CeB23a57C82128bC0c5B2422B63CE1
Ebond.sol	0x266437E6c7500A947012F19A3dE96a3881a0449E
Ethernal.sol	0x17cbd9C274e90C537790C51b4015a65cD015497e
EthernalOracle.sol	0x2885a5b030bB979f7F19c692b3C9F1C7Bc8829E3
EthernalSunrise.sol	0x813c989395f585115152f5D54FdD181fC19CA82a
EthernalTreasury.sol	0x8a7FFE979095F0ee1Fed23B455d25b46d11B6B9D
Lbond.sol	0x3a101bA3f4a39C921A171473592D4EBDA6bD0B57
Lunar.sol	0x25e801Eb75859Ba4052C4ac4233ceC0264eaDF8c
LunarOracle.sol	0xb028D816B5f679c2aABBEa809EA70D523F8c7707
LunarSunrise.sol	0xf3Cd8F422ffE23434C011f43F61879373b31a913
LunarTreasury.sol	0x3A859a07C3b931dcaEC1940c24a4aEfE20D53177
Masonry.sol	0xA452f676F109d34665877B7a7B203f2B445D7DE0
Oracle.sol	0x73E2bBcb511C7De83f489205788304c3Bc31A119
Pbond.sol	0x3a4773e600086A753862621A26a2E3274610da43
Polar.sol	0xf0f3b9Eee32b1F490A4b8720cf6F005d4aE9eA86

Spolar.sol	0x9D6fc90b25976E40adaD5A3EdD08af9ed7a21729
Treasury.sol	0x2Af7e5b19405A02FC99468Af38a23aa270921781
Tribond.sol	0x8200B4F47eDb608e36561495099a8caF3F806198
Tripolar.sol	0x60527a2751A827ec0Adf861EfcAcbf111587d748
TripolarOracle.sol	0xad054C533ea23486eEc6060975eb8B4Ad35fc4E2
TripolarSunrise.sol	0x203a65b3153C55B57f911Ea73549ed0b8EC82B2D
TripolarTreasury	0x116449c7B1d0837439E12216d7b1EbaecbD848FF

Libraries And Interfaces

Interfaces/IBasisAsset.sol
Interfaces/IDecimals.sol
Interfaces/IDistributor.sol
Interfaces/IERC20.sol
Interfaces/ILunarTreasury.sol
Interfaces/IMasonry.sol
Interfaces/IOracle.sol
Interfaces/IShare.sol
Interfaces/ISimpleERCFund.sol
Interfaces/ITaxable.sol
Interfaces/ITreasury.sol
Interfaces/ITripolarTreasury.sol
Interfaces/ITShareRewardPool.sol
Interfaces/IUniswapV2Callee.sol
Interfaces/IUniswapV2ERC20.sol
Interfaces/IUniswapV2Factory.sol
Interfaces/IUniswapV2Pair.sol
Interfaces/IUniswapV2Router.sol
Interfaces/IWrappedFtm.sol
lib/Babylonian.sol
lib/FixedPoint.sol
lib/Safe112.sol
lib/SafeMath8.sol
lib/UniswapV2Library.sol
lib/UniswapV2OracleLibrary.sol
lib/UQ112x112.sol
owner/Operator.sol
utils/ContractGuard.sol
utils/Epoch.sol

Revisions

Revision 1	46ad8aab2592c94d2324d3c074c7fe72bbda2322
Revision 2	d9a7cedd355718014bc2206305e0c7876931b829

Imported Contracts

OpenZeppelin	3.4.2

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