OBELISK

Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 35	2023-01-09	ByFixter, Donut	Audit Final

Audit Notes

Audit Date	2022-11-21 - 2023-01-09
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	OB555546891

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	Level Finance
Description	A Decentralized Perpetual Exchange with Functional Risk Management and Innovative Liquidity Solutions.
Website	https://level.finance/
Contact	https://twitter.com/Level__Finance
Contact information	@sonic_level on TG
Token Name(s)	N/A
Token Short	N/A
Contract(s)	See Appendix A
Code Language	Solidity
Chain	BNB

Audit of Level Core

Obelisk was commissioned by Level Finance on the 6th of November 2022 to conduct a comprehensive audit of levels' Core contracts. The following audit was conducted between the 21st of November 2022 and the 9th of January 2023. 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.

While conducting the audit of Level Finances core contracts, the auditors found multiple issues of varying degrees. The project team worked to close or mitigate most of these issues found with only issues #2 and #5 partially mitigated/closed. Issue #2 is mitigated by transferring ownership to a timelock, however, the current delay is 12 hours while Obelisk recommends 72 hours. The project team says that they would prolong the delay at a later time. Issue #5 relates to a minor rounding error, please see the project comment on this issue.

Issue #12, #13, and #14 are related to our on-chain analysis and all relate to the same problem as issue #5, that the current timelock is set to 12 hours instead of 72 hours or more. The project team says that they would prolong the delay at a later time.

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 Level Finance 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
No Limit For Protocol Values	#0001	High Risk	Closed
emergencyWithdraw can be disabled	#0002	Medium Risk	Partially Mitigated
Rewards Can Be Frozen	#0003	Medium Risk	Closed
External Calls	#0004	Medium Risk	Closed
Division Before Multiplication	#0005	Low Risk	Partially Closed
Governance Not In Use	#0006	Informational	Closed
Event Emited Twice	#0007	Informational	Closed
Unnecessary Argument	#0008	Informational	Closed
Program Can Be Disabled Before End Time	#0009	Low Risk	Closed
Claims Break If Rewarder Has Insufficient Funds	#0011	Medium Risk	Closed

On-Chain Analysis

Finding	ID	Severity	Status
Owner Is An EOA	#0012	High Risk	Partially Mitigated
ProxyAdmin Is Owned By An EOA	#0013	High Risk	Partially Mitigated
Timelock Delay Is Short	#0014	Medium Risk	Partially Mitigated
Contracts Not Initialized	#0015	Medium Risk	Mitigated

Findings

Code Analysis

No Limit For Protocol Values

FINDING ID	#0001
SEVERITY	High Risk
STATUS	Closed
LOCATION	• LevelMaster.sol -> 121-141: *function add(uint256 allocPoint, IERC20 _lpToken, IRewarder _rewarder) public onlyOwner* • LevelMaster.sol -> 148-166: *function set(uint256 _pid, uint256 _allocPoint, IRewarder _rewarder, bool overwrite) public onlyOwner* • LevelStake.sol -> 44-62: *function initialize(address _lvl, address _lgo, uint256 _cooldownSeconds, uint256 _unstakeWindow, uint256 _rewardPerSecond, address _lgoReserve) external initializer* • LevelStake.sol -> 236-240: *function setRewardPerSecond(uint256 _rewardPerSecond) public onlyOwner* • LoyaltyRedeemProgram.sol -> 31-43: *function config(uint256 _startTime, uint256 _endTime, uint256 _conversionRate, uint256 _maxReward) external onlyOwner*

DESCRIPTION	Values can be set arbitrarily high, potentially breaking the functionality of the contracts.
RECOMMENDATION	Add bounds to the values.
RESOLUTION	All variables have had limits added or those stated variables have been removed

emergencyWithdraw can be disabled

FINDING ID	#0002
SEVERITY	Medium Risk
STATUS	Partially Mitigated
LOCATION	LockDrop.sol -> 181-190

```
1     function emergencyWithdraw(address _to) external {
2         require(enableEmergency, "LockDrop::emergencyWithdraw: not in
  emergency");
3
4         uint256 amount = userInfo[msg.sender].amount;
5         if (amount > 0) {
6             delete userInfo[msg.sender];
7             lp.safeTransfer(_to, amount);
8             emit EmergencyWithdrawn(msg.sender, _to);
9         }
10    }
```

DESCRIPTION	The *emergencyWithdraw()* function can be disabled (and is disabled by default).
RECOMMENDATION	Remove this functionality and add a timelock for disabling this functionality. Obelisk recommends a delay of at least 72 hours.
RESOLUTION	Ownership was transferred to a timelock with a delay of 12 hours. Timelock: 0x360071D15cce5542E6B7209752eA479b84b28625 **Project Team Comment:** “After launch, a lot of parameters need to be updated to optimize, so the timelock delay can't be long. We will schedule to increase timelock delay to 24hr, 48hr and 72hr later”

Rewards Can Be Frozen

FINDING ID	#0003
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	Erc20Reserve.sol -> 44-69

```
1     function removeDistributor(address _distributor) external onlyOwner
  {
2         if (!isDistributor[_distributor]) {
3             revert NotDistributor(_distributor);
4         }
5         isDistributor[_distributor] = false;
6         for (uint256 i = 0; i < allDistributors.length; i++) {
7             if (allDistributors[i] == _distributor) {
8                 allDistributors[i] =
  allDistributors[allDistributors.length - 1];
9                 break;
10            }
11        }
12        allDistributors.pop();
13        emit DistributorRemoved(_distributor);
14    }
15
16    function requestTransfer(address _to, uint256 _amount) external {
17        if (!isDistributor[msg.sender]) {
18            revert NotDistributor(msg.sender);
19        }
20        if (_to == address(0)) {
21            revert InvalidAddress(_to);
22        }
23
24        TOKEN.safeTransfer(_to, _amount);
25        emit Distributed(_to, _amount);
26    }
```

DESCRIPTION	If the Distributor is removed no rewards can be withdrawn.
RECOMMENDATION	Remove this function or add a timelock to this function. Obelisk recommends a delay of at least 72 hours.
RESOLUTION	Functionally has been removed.

External Calls

FINDING ID	#0004
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	LevelMaster.sol -> 309-343

```
    function addLiquidity(
        uint256 pid,
        address assetToken,
        uint256 assetAmount,
        uint256 minLpAmount,
        address to
    )
        external
        payable
        nonReentrant
    {
        // ...
        if (assetToken != UniERC20.ETH) {
            // ...
            levelPool.addLiquidity(
                tranche, assetToken, assetAmount, minLpAmount,
address(this)
            );
        } else {
            // ...
            levelPool.addLiquidity{value: assetAmount}(
                tranche, assetToken, assetAmount, minLpAmount,
address(this)
            );
        }
        // ...
    }
```

LOCATION	LockDrop.sol -> 203-216

```
1     function addLiquidity(address _token, uint256 _amount, uint256
  _minLpAmount) internal returns (uint256) {
2         // ...
3
4         if (_token != UniERC20.ETH) {
5             // ...
6             pool.addLiquidity(address(lp), _token, _amount,
  _minLpAmount, address(this));
7         } else {
8             // ...
9             pool.addLiquidity{value: _amount}(address(lp), _token,
  _amount, _minLpAmount, address(this));
10        }
11
12        // ...
13    }
```

DESCRIPTION	Contracts make calls to *pool.addLiquidity*. See Level-trading audit that includes this contract.
RECOMMENDATION	N/A
RESOLUTION	**Project team comment:** - We fixed the *redeemCooldown* bypass/DoS, and wrap all ETH regarding the changes we made in trading contracts.

Division Before Multiplication

FINDING ID	#005
SEVERITY	Low Risk
STATUS	Partially Closed
LOCATION	LevelMaster.sol ->179-199

```
    function pendingReward(uint256 _pid, address _user)
        external
        view
        returns (uint256 pending)
    {
        // ...
        if (block.timestamp > pool.lastRewardTime && lpSupply != 0) {
            uint256 time = block.timestamp - pool.lastRewardTime;
            uint256 reward =
                time * rewardPerSecond * pool.allocPoint / totalAllocPoint;
            accRewardPerShare =
                accRewardPerShare + (reward * ACC_REWARD_PRECISION / lpSupply);
        }
        pending = uint256(
            int256(user.amount * accRewardPerShare / ACC_REWARD_PRECISION)
                - user.rewardDebt
        );
    }
```

LOCATION	LevelMaster.sol -> 213-230

```
1     function updatePool(uint256 pid) public returns (PoolInfo memory
  pool) {
2         // ...
3         if (block.timestamp > pool.lastRewardTime) {
4             // ...
5             if (lpSupply > 0) {
6                 // ...
7                 uint256 reward =
8                     time * rewardPerSecond * pool.allocPoint /
  totalAllocPoint;
9                 pool.accRewardPerShare = pool.accRewardPerShare
10                    + uint128(reward * ACC_REWARD_PRECISION /
  lpSupply);
11            }
12            // ...
13        }
14    }
```

DESCRIPTION	The calculations noted use mixed orders of multiplication and division. This may cause rounding errors, resulting in reverted transactions or miscalculations in general.
RECOMMENDATION	Change the calculations to first multiply, then divide.
RESOLUTION	**Project team comment:** - The *ACC_REWARD_PRECISION* helps keep the accumulated value accurate. We think the rounding error is minor.

Governance Not In Use

FINDING ID	#0006
SEVERITY	Informational
STATUS	Closed
LOCATION	LevelGovernance.sol -> 25-28

```
1     /// @dev reference to the Level governance contract to call (if
  initialized) on _beforeTokenTransfer
2     /// !!! IMPORTANT The Level governance is considered a trustable
  contract, being its responsibility
3     /// to control all potential reentrancies by calling back the
  LevelToken
4     ITransferHook public _levelGovernance;
```

LOCATION	LevelGovernance.sol -> 110-114

```
1         // caching the level governance address to avoid multiple state
  loads
2         ITransferHook levelGovernance = _levelGovernance;
3         if (levelGovernance != ITransferHook(address(0))) {
4             levelGovernance.onTransfer(from, to, amount);
5         }
```

LOCATION	GovernancePowerWithSnapshot.sol -> 27-29

```
1     function _setLevelGovernance(ITransferHook levelGovernance)
  internal virtual {
2         _levelGovernance = levelGovernance;
3     }
```

DESCRIPTION	The set function is an internal function that is never called meaning this functionality is not currently being used.
RECOMMENDATION	If the transfer hook does not need to be called on the governance contract, remove the functionality and associated variables. Otherwise, add the external method (restricted access) to set the governance contract.

RESOLUTION	Functions/contracts have been removed.

Event Emited Twice

FINDING ID	#0007
SEVERITY	Informational
STATUS	Closed
LOCATION	LevelMaster.sol -> 236-240

```
    function deposit(uint256 pid, uint256 amount, address to) public {
        _deposit(pid, amount, to);
        lpToken[pid].safeTransferFrom(msg.sender, address(this),
amount);
        emit Deposit(msg.sender, pid, amount, to);
    }
```

LOCATION	LevelMaster.sol -> 382-397

```
    function _deposit(uint256 pid, uint256 amount, address to)
internal {
        PoolInfo memory pool = updatePool(pid);
        UserInfo storage user = userInfo[pid][to];

        // Effects
        user.amount = user.amount + amount;
        user.rewardDebt = user.rewardDebt
            + int256(amount * pool.accRewardPerShare /
ACC_REWARD_PRECISION);

        // Interactions
        IRewarder _rewarder = rewarder[pid];
        if (address(_rewarder) != address(0)) {
            _rewarder.onReward(pid, msg.sender, to, 0, user.amount);
        }
        emit Deposit(msg.sender, pid, amount, to);
    }
```

DESCRIPTION	The *deposit()* function emits the *Deposit* event twice.
RECOMMENDATION	Remove one of the emits.
RESOLUTION	The event is now only emitted once.

Unnecessary Argument

FINDING ID	#0008
SEVERITY	Informational
STATUS	Closed
LOCATION	LevelStake.sol -> 197-230

```
1    function getNextCooldownTimestamp(
2        uint256 _fromCooldownTimestamp,
3        uint256 _amountToReceive,
4        address _to,
5        uint256 _toBalance
6    )
7        public
8        view
9        returns (uint256)
10   {
11       // ...
12   }
```

DESCRIPTION	Since LevelStake is non-transferrable *getNextCooldownTimestamp()* is only ever called with the argument *_fromCooldownTimestamp == 0*. This makes the argument unnecessary.
RECOMMENDATION	Remove the argument *_fromCooldownTimestamp*.
RESOLUTION	Argument has been removed.

Program Can Be Disabled Before End Time

FINDING ID	#0009
SEVERITY	Low Risk
STATUS	Closed
LOCATION	Rev-2 LoyaltyRedeemProgram.sol -> 40-92

```
1     function addProgram(
2         uint256 _startTime,
3         uint256 _endTime,
4         uint256 _conversionRate,
5         uint256 _maxReward,
6         bool isActive
7     ) external onlyOwner {
8         // ...
9         if (isActive) {
10            activeProgramId = nextProgramId;
11        }
12        // ...
13    }
14
15    /// @notice burn lyLVL to get an amount of LVL
16    /// the conversion rate is fixed when config
17    function redeem(address _to, uint256 _amount) external
  whenNotPaused nonReentrant {
18        // ...
19    }
20
21    function activeProgram(uint256 _pid) external onlyOwner {
22        activeProgramId = _pid;
23        emit ProgramActived(_pid);
24    }
25
26    function pause() external onlyOwner {
27        _pause();
28    }
29
30    function unpause() external onlyOwner {
31        _unpause();
32    }
```

DESCRIPTION	The active program can be changed or paused before its end time by the owner. This can lead to users not being able to withdraw from that program.
RECOMMENDATION	Inform the end user in the front end.
RESOLUTION	LoyaltyRedeemProgram.sol has been removed.

Claims Break If Rewarder Has Insufficient Funds

FINDING ID	#0011
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	rev-3 tokens/LyLevel.Sol->199-211

```
    /// @notice allocate reward for current batch and start a new batch
    function allocateReward(uint256 _totalAmount) external onlyOwner {
        require(totalSupply() > 0, "LyLevel:no supply");
        RedeemProgramInfo memory info = RedeemProgramInfo({
            totalBalance: totalSupply(),
            rewardPerShare: _totalAmount * PRECISION / totalSupply(),
            allocatedTime: block.timestamp
        });
        redeemPrograms[currentBatchId] = info;
        emit RewardAllocated(currentBatchId, _totalAmount);
        currentBatchId++;
        emit BatchStarted(currentBatchId);
    }
```

LOCATION	rev-3 tokens/LyLevel.Sol->55-59

```
    function claimable(uint256 _batchId, address _account) public view
returns (uint256) {
        require(_batchId <= currentBatchId, "LyLevel: program not
exist");
        return userBalance[_batchId][_account] *
redeemPrograms[_batchId].rewardPerShare / PRECISION
            - userClaimed[_batchId][_account];
    }
```

LOCATION	rev-3 tokens/LyLevel.Sol->117-125

```
    function claim(uint256 _batchId, address _receiver) external {
        require(address(rewardFund) != address(0), "LyLevel: reward
fund not set");
        address sender = _msgSender();
        uint256 amount = claimable(_batchId, sender);
        require(amount != 0, "LyLevel: nothing to claim");
        userClaimed[_batchId][sender] += amount;
        rewardFund.requestTransfer(_receiver, amount);
        emit Claimed(sender, _batchId, amount, _receiver);
    }
```

DESCRIPTION	If the rewarder does not contain *_totalAmount* of rewards, the rewards will be unevenly distributed based on who claims first. Early claims will exhaust all of the available rewards and later claimers will not be able to claim their fair share.
RECOMMENDATION	Check that *_totalAmount* unallocated rewards are available in the rewarder contract by tracking the balances in the rewarder contract whenever new rewards are added or rewards are allocated to the *LyLevel.Sol* contract. When *allocateReward* is called in *LyLevel.Sol*, the rewarder contract can check that it has enough unallocated funds available.
RESOLUTION	The project team has implemented the recommended changes.

On-Chain Analysis

Owner Is An EOA

FINDING ID	#0012
SEVERITY	High Risk
STATUS	Partially Mitigated
LOCATION	LockDrop 0xd804ea7306abe2456bdd04a31f6f6a2f55dc0d21 LyLevel 0xf6F99E15E0ac60cAB6E3d2B4Bfe6B26BF654Bcd9

DESCRIPTION	The owner of the contracts is an EOA and can arbitrarily change parameters that affect how the contract functions.
RECOMMENDATION	Change the owner to a timelock. Obelisk recommends a delay of at least 72 hours.
RESOLUTION	Ownership was transferred to a timelock with a delay of 12 hours. Timelock: 0x360071D15cce5542E6B7209752eA479b84b28625 **Project Team Comment:** “After launch, a lot of parameters need to be updated to optimize, so the timelock delay can't be long. We will schedule to increase timelock delay to 24hr, 48hr and 72hr later”

ProxyAdmin Is Owned By An EOA

FINDING ID	#0013
SEVERITY	High Risk
STATUS	Partially Mitigated
LOCATION	LevelGovernance 0xBe2B6C5E31F292009f495DDBda88e28391C9815E LevelStake 0x87CC04d6FE59859cB7eB6d970EBc22dCdCBc9368 LyLevel 0xf6F99E15E0ac60cAB6E3d2B4Bfe6B26BF654Bcd9 ProxyAdmin 0x8f886b4b10344289cEAd777953f95FA0317bcD33

DESCRIPTION	The ProxyAdmin owner is an EOA and can upgrade the contracts managed by the ProxyAdmin. This can result in arbitrary changes to the functionality of the contracts.
RECOMMENDATION	Change the owner to a timelock. Obelisk recommends a delay of at least 72 hours.
RESOLUTION	Ownership was transferred to a timelock with a delay of 12 hours. Timelock: 0x360071D15cce5542E6B7209752eA479b84b28625 **Project Team Comment:** “After launch, a lot of parameters need to be updated to optimize, so the timelock delay can't be long. We will schedule to increase timelock delay to 24hr, 48hr and 72hr later” Note: LyLevel still pending

Timelock Delay Is Short

FINDING ID	#0014
SEVERITY	Medium Risk
STATUS	Partially Mitigated
LOCATION	LevelMaster 0x1Ab33A7454427814a71F128109fE5B498Aa21E5d LevelStake 0x87CC04d6FE59859cB7eB6d970EBc22dCdCBc9368 Timelock 0x360071D15cce5542E6B7209752eA479b84b28625

DESCRIPTION	The timelock contract has a 12 hour delay.
RECOMMENDATION	Obelisk recommends a delay of at least 72 hours.
RESOLUTION	**Project Team Comment:** “After launch, a lot of parameters need to be updated to optimize, so the timelock delay can't be long. We will schedule to increase timelock delay to 24hr, 48hr and 72hr later”

Contracts Not Initialized

FINDING ID	#0015
SEVERITY	Medium Risk
STATUS	Mitigated
LOCATION	LockDrop 0xd804ea7306abe2456bdd04a31f6f6a2f55dc0d21 LyLevel 0xf6F99E15E0ac60cAB6E3d2B4Bfe6B26BF654Bcd9

DESCRIPTION	LockDrop - levelMaster: is zero address LyLevel - rewardToken: is zero address - minter: is zero address
RECOMMENDATION	Initialize the contract to set these addresses.
RESOLUTION	The values were initialized

External Addresses

Externally Owned Accounts

Owner

ACCOUNT	0xDC8Ee58cebFF504cCf6222c55A8F27f6033DfeC4
USAGE	0x804bbb7a06c0934571aAD137360215ef1335e6A1 *LevelDevFund.owner* - Variable 0xf6F99E15E0ac60cAB6E3d2B4Bfe6B26BF654Bcd9 *LyLevel.owner* - Variable
IMPACT	• receives elevated permissions as owner, operator, or other

External Contracts

These contracts are not part of the audit scope.

Level Pool

ADDRESS	0xA5aBFB56a78D2BD4689b25B8A77fd49Bb0675874
USAGE	0xd804ea7306abe2456bdd04a31f6f6a2f55dc0d21 *LockDrop.pool* - Immutable 0x1Ab33A7454427814a71F128109fE5B498Aa21E5d *LevelMaster.levelPool* - Immutable
IMPACT	• receives allowance of tokens deposited by users • impacts ability to deposit or withdraw tokens

External Tokens

These contracts are not part of the audit scope.

Wrapped BNB

ADDRESS	0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
USAGE	0xd804ea7306abe2456bdd04a31f6f6a2f55dc0d21 *LockDrop.weth* - Immutable 0x1Ab33A7454427814a71F128109fE5B498Aa21E5d *LevelMaster.weth* - Immutable
IMPACT	• ERC20 Token

Mezzanine LLP

ADDRESS	0x4265af66537F7BE1Ca60Ca6070D97531EC571BDd
USAGE	0xd804ea7306abe2456bdd04a31f6f6a2f55dc0d21 *LockDrop.lp* - Immutable
IMPACT	• ERC20 Token

Appendix A - Reviewed Documents

Deployed Contracts

Document	Address
farm/LevelMaster.sol	0x1Ab33A7454427814a71F128109fE5B498Aa21E5d
farm/LevelStake.sol	Proxy 0x87CC04d6FE59859cB7eB6d970EBc22dCdCBc9368 Implementation 0x71d5b2B736726A1524F09163aCd29416eB46b4E4
fund/Erc20Reserve.sol	0x92A0A11A57C28d4C86a629530fd59B83B1276003 0x9a1409a1b7826A80B6C6D33f85a342Cd9448FB54
fund/LevelDevFund.sol	0x804bbb7a06c0934571aAD137360215ef1335e6A1
lockdrop/LockDrop.sol	0xd804ea7306abe2456bdd04a31f6f6a2f55dc0d21
loyalty/LoyaltyRedeemProgram.sol	N/A
timelock/Timelock.sol	12 hour delay 0x360071D15cce5542E6B7209752eA479b84b28625 72 hour delay 0xa0E6B7aE9F4D284df33D41B1355aC28CBDC64F8c
tokens/LevelGovernance.sol	Proxy 0xBe2B6C5E31F292009f495DDBda88e28391C9815E Implementation 0xB78F9cC4ebDF4D68DC539315275002867946050c
tokens/LevelToken.sol	0xB64E280e9D1B5DbEc4AcceDb2257A87b400DB149
tokens/LyLevel.sol	Proxy 0x95883611685a20936EC935B0A33F82e11D478e3D Implementation 0x5534b399edaebaf07be46E1C5B36e8Bc0b4859E2
OpenZeppelin ProxyAdmin	0x8f886b4b10344289cEAd777953f95FA0317bcD33

Libraries And Interfaces

fund/Erc20Reserve.sol fund/TokenVesting.sol interfaces/IGovernancePowerDelegationToken.sol interfaces/ILevelMaster.sol interfaces/ILevelStake.sol interfaces/ILPToken.sol interfaces/ILyLevelToken.sol interfaces/IPool.sol interfaces/IRewarder.sol interfaces/ITokenReserve.sol interfaces/IWETH.sol lib/GovernancePowerDelegationERC20.sol

Revisions

Revision 1	9dae5a31b2749620e67d66bf48eec2a6a0143493
Revision 2	15ea3d3a67c82ad5b7555375de5858c74cc6bf55
Revision 3	ab437cdc77866684c27fa9c28ed1c091a379fdd2
Revision 4	01d93965356d48a5b5a705975c6d8ebbca85cb53

Imported Contracts

OpenZeppelin	4.8.0

Appendix B - Risk Ratings

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

Appendix C - Finding Statuses

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

Appendix D - Glossary

Contract Structure

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

Security Concepts

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

Appendix E - Audit Procedure

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

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

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

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

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

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

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

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

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

Follow Obelisk Auditing for the Latest Information

 ObeliskOrg

 ObeliskOrg

OBELISK

Part of Tibereum Group