OBELISK

Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 69	2022-03-28	Donut, Plemonade	Audit Final

Audit Notes

Audit Date	2022-02-15 - 2022-03-27
Auditor/Auditors	Plemonade, 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	OB55896258

Disclaimer

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

Obelisk Auditing

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

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

Audit Information

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

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

Table of Contents

Project Information

Name	BASED
Description	Based Finance is launched with the main purpose of supporting Tomb Finance by pegging BASED to TOMB, thus giving additional use cases to TOMB as well as attracting more TVL into the Fantom ecosystem. The BASED algorithmic token aims to serve as the backbone of a rapidly growing Fantom ecosystem and aims towards bringing new liquidity.
Website	https://basedfinance.io/
Contact	@BasedFinance_io on Twitter
Contact information	@athena_goddazz on TG
Token Name(s)	N/A
Token Short	N/A
Contract(s)	See Appendix A
Code Language	Solidity
Chain	Fantom

Audit of Based

Obelisk was commissioned by Based on the 12th of February 2022 to conduct a comprehensive audit of Baseds' contracts. The following audit was conducted between the 15th of February 2022 and the 27th of March 2022. Two of Obelisk's security experts went through the related contracts manually using industry standards to find if any vulnerabilities could be exploited either by the project team or users.

The audit was conducted in two stages, first on the GitHub repository and then on deployed contracts. The deployed contracts differed from the repository, and many of the issues found were fixed in GitHub but without corresponding fixes on chain. The issues marked closed or mitigated were fixed on-chain, or via newly deployed contracts (such as the zapper). A significant number of the outstanding issues would require a full re-deployment and migration to resolve. Please read the comments on each issue found to get a full understanding of the impact on the protocol.

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

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

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

Summary Table

Manual Analysis

Finding	ID	Severity	Status
No Price Impact Or Slippage Protection	#0001	High Risk	Closed
Stale Price 1	#0002	High Risk	Open
Stale Price 2	#0003	High Risk	Open
Unsupported Token Recovery Is Unrestricted After Pools Expire	#0004	Medium Risk	Partially Mitigated
Reentrancy Risk Using Token With Callbacks	#0005	Medium Risk	Mitigated
Initalize Can Be Called By Anyone	#0006	Medium Risk	Mitigated
Contract Accumulates Leftover Swap Tokens	#0007	Medium Risk	Closed
ERC20 Tokens Can Be Claimed By Anyone	#0008	Medium Risk	Closed
Use Of Liquidity Pair Balance For Prices	#0009	Medium Risk	Closed
Team Can Add New Team Pool	#0010	Low Risk	Open
UniswapV2Library Contains Network Dependent Code	#0011	Informational	Open
No Lower Limit For Lockup Epochs	#0012	Informational	Open
No Limits For Discount rate	#0013	Informational	Open
Comments Do Not Match Contract Code	#0014	Informational	Open
Unnecessary Override	#0015	Informational	Open
Unused Modifier	#0016	Informational	Closed
Confusing Token Name	#0017	Informational	Open
Repetitive Swap Functions Code	#0018	Informational	Open

Incorrect Arguments For _dustDistribution	#0020	Medium Risk	Open
IZapper Interface Does Not Match Zapper	#0021	Medium Risk	Closed
Oracle May Drift	#0022	Informational	Open
DoS Can Occur With Unexpected Revert	#0023	High Risk	Open
_swap Does Not Approve Or Transfer Tokens	#0024	Low Risk	Closed
Burn Address Can Be Updated	#0025	Low Risk	Open
Updating Native Token Breaks Swap Paths	#0026	Low Risk	Mitigated
Fixed Swap Path Lengths	#0027	Informational	Mitigated
Zapper Might Use Malicious Liquidity Pools	#0028	Informational	Mitigated
Router Trust	#0030	Informational	Open

Static Analysis

Finding	ID	Severity	Status
Division Before Multiplication 1	#0019	Informational	Open
Division Before Multiplication 2	#0029	Low Risk	Open

On-Chain Analysis

Finding	ID	Severity	Status
No Timelock	#0031	High Risk	Open
Multisig Low Amount Of Signers	#0032	Informational	Open

Findings

Manual Analysis

No Price Impact Or Slippage Protection

FINDING ID	#0001
SEVERITY	High Risk
STATUS	Closed
LOCATION	BasedTombZap.sol -> 46-52 BShareFtmZap.sol -> 47-53

```
1     function zapInToken(address _from, uint256 amount, address _to,
  address routerAddr, address _recipient) external {
2         // From an ERC20 to an LP token, through specified router,
  going through base asset if necessary
3         IERC20(_from).safeTransferFrom(msg.sender, address(this),
  amount);
4         // we'll need this approval to add liquidity
5         _approveTokenIfNeeded(_from, routerAddr);
6         _swapTokenToLP(_from, amount, _to, _recipient, routerAddr);
7     }
```

DESCRIPTION	The zapper contracts do not check for slippage on swaps and can be vulnerable to price manipulation. A user can be front-run if swapping a large amount or be affected by a very high price impact without noticing. Additionally, a malicious router can cause users to lose all funds. A high price impact can also result in the loss of some output tokens (refer to finding 7).
RECOMMENDATION	Add price impact/slippage protection. For a swap and zapper contract, this is typically done by: • checking the spot price in the frontend • allowing the user to choose a slippage limit • calculating a minimum amount for the user to receive • passing the minimum amount to the zapper

RESOLUTION	All callers now check that the return value from the router on an LP or Swap operation exceeds the user-supplied minimum.

Stale Price 1

FINDING ID	#0002
SEVERITY	High Risk
STATUS	Open
LOCATION	Oracle.sol -> 62-81

```
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    }
```

LOCATION	Treasury.sol -> 412-414

```
1     function _updateBasedPrice() internal {
2         try IOracle(basedOracle).update() {} catch {}
3     }
```

DESCRIPTION	A Uniswap LP oracle is expected to make use of overflowing to deal with cumulative prices. However, from

	solidity 0.8.0, basic arithmetic operations automatically revert on overflow. If the update fails because of this, the oracle price will no longer update as expected and the treasury will operate with a stale price.
RECOMMENDATION	Add unchecked to the oracle logic where expected.
RESOLUTION	**Deployed Contracts:** Open **Rev-5 Contracts:** Closed The project team has implemented the recommended fix. **Project team comment:** “We are talking to chainlink to use their oracles and all price feeds, if we don’t get them in a short amount of time - will redeploy contracts with overflow fixed”

Stale Price 2

FINDING ID	#0003
SEVERITY	High Risk
STATUS	Open
LOCATION	Oracle.sol -> 84-91

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

LOCATION	Treasury.sol -> 163-169

```
    function getBasedPrice() public view returns (uint256 basedPrice)
{
        try IOracle(basedOracle).consult(based, 1e18) returns (uint144 price) {
            return uint256(price);
        } catch {
            revert("Treasury: failed to consult BASED price from the oracle");
        }
    }
```

LOCATION	Treasury.sol -> 298-300

```
    function setBasedOracle(address _basedOracle) external onlyOperator {
        basedOracle = _basedOracle;
    }
```

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()*.
RECOMMENDATION	Ensure that the oracle is always in a correct state when calling it.
RESOLUTION	**Deployed Contracts:** Open **Rev-5 Contracts:** Closed The project team has implemented the recommended fix. **Project team comment:** “We are talking to chainlink to use their oracles and all price feeds, if we don’t get them in a short amount of time - will redeploy contracts with overflow fixed”

Unsupported Token Recovery Is Unrestricted After Pools Expire

FINDING ID	#0004
SEVERITY	Medium Risk
STATUS	Partially Mitigated
LOCATION	distribution/BasedGenesisRewardPool.sol -> 279-290

```
    function governanceRecoverUnsupported(IERC20 _token, uint256 amount, address to) external onlyOperator {
        if (block.timestamp < poolEndTime + 3 days) {
            // do not allow to drain core token (BASED or lps) if less than 3 days after pool ends
            require(_token != based, "based");
            uint256 length = poolInfo.length;
            for (uint256 pid = 0; pid < length; ++pid) {
                PoolInfo storage pool = poolInfo[pid];
                require(_token != pool.token, "pool.token");
            }
        }
        _token.safeTransfer(to, amount);
    }
```

LOCATION	distribution/BShareRewardPool.sol -> 287-298

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

DESCRIPTION	After the governance pools have closed, the operator will be able to transfer any remaining tokens out after a period of days (spanning from 3 to 90 days) to an arbitrary address.

RECOMMENDATION	Remove the conditional such that users' pool tokens can never be recovered from the pool.
RESOLUTION	**Deployed Contracts:** Partially Mitigated BShareRewardPool operator is still able to recover deposits. 0xAc0fa95058616D7539b6Eecb6418A68e7c18A746 BasedGenesisRewardPool operator has been renounced. 0x9Ec66B9409d4cD8D4a4C90950Ff0fd26bB39ad84 **Rev-5 Contracts:** Closed The project team removed the *governanceRecoverUnsupported()* function. **Project team comment:** "bshareRewardPool contract will be renounced as soon as we have the right amount of pools and needed allocation points for each."

Reentrancy Risk Using Token With Callbacks

FINDING ID	#0005
SEVERITY	Medium Risk
STATUS	Mitigated
LOCATION	distribution/BasedGenesisRewardPool.sol -> 183-203

```
1     function updatePool(uint256 _pid) public {
2         PoolInfo storage pool = poolInfo[_pid];
3         if (block.timestamp <= pool.lastRewardTime) {
4             return;
5         }
6         uint256 tokenSupply = pool.token.balanceOf(address(this));
7         if (tokenSupply == 0) {
8             pool.lastRewardTime = block.timestamp;
9             return;
10        }
11        if (!pool.isStarted) {
12            pool.isStarted = true;
13            totalAllocPoint = totalAllocPoint.add(pool.allocPoint);
14        }
15        if (totalAllocPoint > 0) {
16            uint256 _generatedReward =
  getGeneratedReward(pool.lastRewardTime, block.timestamp);
17            uint256 _basedReward =
  _generatedReward.mul(pool.allocPoint).div(totalAllocPoint);
18            pool.accBasedPerShare =
  pool.accBasedPerShare.add(_basedReward.mul(1e18).div(tokenSupply));
19        }
20        pool.lastRewardTime = block.timestamp;
21    }
```

LOCATION	distribution/BasedGenesisRewardPool.sol -> 207-229

```
    function deposit(uint256 _pid, uint256 _amount) public {
        address _sender = msg.sender;
        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][_sender];
        updatePool(_pid);
        if (user.amount > 0) {
            // transfer rewards to user if any pending rewards
            uint256 _pending =
user.amount.mul(pool.accBasedPerShare).div(1e18).sub(user.rewardDebt)
;
            if (_pending > 0) {
                // send pending reward to user, if rewards
accumulating in _pending
                safeBasedTransfer(_sender, _pending);
                emit RewardPaid(_sender, _pending);
            }
        }
        if (_amount > 0) {
            pool.token.safeTransferFrom(_sender, address(this),
_amount);
            uint256 depositDebt = _amount.mul(50).div(10000);
            user.amount = user.amount.add(_amount.sub(depositDebt));
            pool.token.safeTransfer(daoFundAddress, depositDebt);
        }
        user.rewardDebt =
user.amount.mul(pool.accBasedPerShare).div(1e18);
        emit Deposit(_sender, _pid, _amount);
    }
```

LOCATION	distribution/BasedGenesisRewardPool.sol -> 233-250

```
    function withdraw(uint256 _pid, uint256 _amount) public {
        address _sender = msg.sender;
        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][_sender];
        require(user.amount >= _amount, "withdraw: not good");
        updatePool(_pid);
        uint256 _pending =
user.amount.mul(pool.accBasedPerShare).div(1e18).sub(user.rewardDebt)
;
        if (_pending > 0) {
            safeBasedTransfer(_sender, _pending);
            emit RewardPaid(_sender, _pending);
        }
        if (_amount > 0) {
            user.amount = user.amount.sub(_amount);
            pool.token.safeTransfer(_sender, _amount);

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

DESCRIPTION	If a token has callbacks (such as an *onReceived* function in the standard ERC721), then a malicious actor may be able to drain the contract of tokens either via the withdraw or deposit function. Note that *ContractGuard* does not prevent reentrancy.
RECOMMENDATION	Follow the checks-effects-interactions pattern. Alternatively, add a reentrancy guard.
RESOLUTION	**Deployed Contracts:** Mitigated BasedGenesisRewardPool has ended. **Rev-5 Contracts:** Closed Both contracts now make use of ReentrancyGuard. **Project team comment:** “Contract is already expired, and there are no funds left in genesis reward pool. Our code base has the safety implemented already for future use.“

Initialize Can Be Called By Anyone

FINDING ID	#0006
SEVERITY	Medium Risk
STATUS	Mitigated
LOCATION	Treasury.sol -> 248-288

```
function initialize(
    address _based,
    address _bbond,
    address _bshare,
    address _basedOracle,
    address _acropolis,
    uint256 _startTime
) public notInitialized {
    // ...
}
```

LOCATION	Acropolis.sol -> 126-144

```
function initialize(
    IERC20 _based,
    IERC20 _share,
    ITreasury _treasury
) public notInitialized {
    // ...
}
```

DESCRIPTION	Anyone can initialize the contract and become the operator if the deployer does not initialize in the same transaction.
RECOMMENDATION	If the Treasury or Acropolis need to be changed make sure that the initializer is protected.
RESOLUTION	**Deployed Contracts:** Mitigated Contracts are initialized. **Rev-5 Contracts:** Closed The project team has implemented the recommendation.

Contract Accumulates Leftover Swap Tokens

FINDING ID	#0007
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	BasedTombZap.sol -> 218 BasedTombZap.sol -> 223

```
1            return router.addLiquidity(token0, token1, token0Amount,
  amount.sub(swapValue), 0, 0, recipient, block.timestamp);
```

LOCATION	BShareFTMZap.sol -> 218

```
1            return router.addLiquidity(token0, token1,
  amount.sub(swapValue), token1Amount, 0, 0, recipient,
  block.timestamp);
```

LOCATION	BShareFTMZap.sol -> 226

```
1            return router.addLiquidityETH{value :
  amount.sub(swapValue)}(token, tokenAmount, 0, 0, recipient,
  block.timestamp);
```

LOCATION	BasedTombZap.sol -> 190 BShareFTMZap.sol -> 192

```
1            ( , , liquidity) =
  IUniswapV2Router(routerAddr).addLiquidity(_from, other,
  amount.sub(sellAmount), otherAmount, 0, 0, recipient,
  block.timestamp);
```

DESCRIPTION	When adding liquidity, 50% of the input tokens are swapped to the paired token. The value of tokens received from the swap will be slightly less anticipated due to slippage and swap fees.

	As a result, when adding liquidity, some of the tokens will be left in the contract. This is usually a small amount, but maybe relatively large for low liquidity pools. Example: ● Start with 10 FOO ($100) ● Swap 5 FOO for BAR ● Receive 4 BAR ($40) ● Add equal amounts of FOO and BAR to LP ($40 FOO and $40 BAR) ● $10 FOO remains in contract
RECOMMENDATION	Return the dust to the user after adding liquidity.
RESOLUTION	The project team has implemented the recommended fix.

ERC20 Tokens Can Be Claimed By Anyone

FINDING ID	#0008
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	BasedTombZap.sol -> 172-176 BShareFTMZap.sol -> 174-178

```
1     function _approveTokenIfNeeded(address token, address router)
  private {
2         if (IERC20(token).allowance(address(this), router) == 0) {
3             IERC20(token).safeApprove(router, type(uint256).max);
4         }
5     }
```

DESCRIPTION	The zapper contracts completely trust the routers it uses for swaps. Because any router can be passed to the contract, malicious contracts can be passed as a parameter. Because the contract gives maximum allowance to the router, a malicious router can later be used to drain any stuck tokens.
RECOMMENDATION	Have whitelisted routers. Only approve the exact swap amounts.
RESOLUTION	The project team implemented whitelisted routers.

Use Of Liquidity Pair Balance For Prices

FINDING ID	#0009
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	BShareSwapper.sol -> 85-89

```
1     function getBasedPrice() public view returns (uint256) {
2         return IERC20(wftmAddress).balanceOf(basedSpookyLpPair)
3         .mul(1e18)
4         .div(based.balanceOf(basedSpookyLpPair));
5     }
```

LOCATION	BShareSwapper.sol -> 91-95

```
1     function getBSharePrice() public view returns (uint256) {
2         return IERC20(wftmAddress).balanceOf(bshareSpookyLpPair)
3         .mul(1e18)
4         .div(bshare.balanceOf(bshareSpookyLpPair));
5     }
```

LOCATION	BShareSwapper.sol -> 97-109

```
1     function getBShareAmountPerBased() public view returns (uint256)
  {
2         uint256 basedPrice =
  IERC20(wftmAddress).balanceOf(basedSpookyLpPair)
3         .mul(1e18)
4         .div(based.balanceOf(basedSpookyLpPair));
5
6         uint256 bsharePrice =
7         IERC20(wftmAddress).balanceOf(bshareSpookyLpPair)
8         .mul(1e18)
9         .div(bshare.balanceOf(bshareSpookyLpPair));
10
11
12        return basedPrice.mul(1e18).div(bsharePrice);
13    }
```

DESCRIPTION	The BShareSwapper uses the *balanceOf()* function on the

	token to determine the state of the liquidity pair instead of the *getReserves()* function. Liquidity pairs use internal bookkeeping to track token balances. The attacker can manipulate the contract balances with flashloans as *balanceOf()* on the token will check the current amount on the pair and not the reserve(the pair contract is currently lending out funds to the attacker). The attacker could also manually manipulate the price without a flashloan and take some risk even if *getReserves()* was used because the contract uses spot pricing. An attacker can then manipulate the price and return it to normal after attacking. However, this would involve risk for the attacker as anyone could return the price to normal while the attack is happening. Manipulating price may allow an attacker to drain the entire swapper.
RECOMMENDATION	Use an oracle based on the liquidity pair's reserves to acquire a steadier price feed.
RESOLUTION	The project team has implemented the recommendation.

Team Can Add New Team Pool

FINDING ID	#0010
SEVERITY	Low Risk
STATUS	Open
LOCATION	BShareRewardPool.sol -> 135-153

```
1     function set(uint256 _pid, uint256 _allocPoint) public
  onlyOperator {
2         massUpdatePools();
3         require (_pid != 2, "CAN NOT ADJUST TEAM ALLOCATIONS");
4
5         PoolInfo storage pool = poolInfo[_pid];
6
7         if (_pid == 0 || _pid == 1) {
8             require(_allocPoint >= 12000 * 10**18, "out of range");
  // >= allocations for lp pools cant be less than 12,000
9             if (pool.isStarted) {
10                totalAllocPoint =
  totalAllocPoint.sub(pool.allocPoint).add(_allocPoint);
11            }
12        } else if (_pid > 2) {
13            if (pool.isStarted) {
14                totalAllocPoint =
  totalAllocPoint.sub(pool.allocPoint).add(_allocPoint);
15            }
16
17        }
18        pool.allocPoint = _allocPoint;
19    }
```

DESCRIPTION	A *require* statement is used to prevent the operator from modifying the team's allocation pool. However, the operator can deploy a new token and pool and effectively bypass this restriction.
RECOMMENDATION	Put a limit on adding a new pool. Otherwise, remove the *require* statement as redundant.
RESOLUTION	**Deployed Contracts:** Open **Rev-5 Contracts:** Partially Closed Allocations have a maximum value that can be set during update, but there is no limit during pool creation. **Project team comment:** “We do add new pools and will

	be changing things around, needed safety features are introduced”

UniswapV2Library Contains Network Dependent Code

FINDING ID	#0011
SEVERITY	Informational
STATUS	Open
LOCATION	lib/UniswapV2Library.sol -> 16-35

```
1      // calculates the CREATE2 address for a pair without making any
  external calls
2      function pairFor(
3          address factory,
4          address tokenA,
5          address tokenB
6      ) internal pure returns (address pair) {
7          (address token0, address token1) = sortTokens(tokenA,
  tokenB);
8          pair = address(
9              uint256(
10                 keccak256(
11                     abi.encodePacked(
12                         hex"ff",
13                         factory,
14                         keccak256(abi.encodePacked(token0, token1)),
15
  hex"96e8ac4277198ff8b6f785478aa9a39f403cb768dd02cbee326c3e7da348845f"
  // init code hash
16                     )
17                 )
18             )
19         );
20     }
```

DESCRIPTION	The init code hash used in the *pairFor* function is specific to the Uniswap on Ethereum mainnet and will not correctly calculate pairs when deployed on other networks.
RECOMMENDATION	Update the library to be compatible with the DEX and chain it will be deployed to.
RESOLUTION	**Project team comment:** “we deployed on FTM network, future network deployments will be addressed accordingly”

No Lower Limit For Lockup Epochs

FINDING ID	#0012
SEVERITY	Informational
STATUS	Open
LOCATION	Acropolis.sol -> 150-153

```
1     function setLockUp(uint256 _withdrawLockupEpochs, uint256
  _rewardLockupEpochs) external onlyOperator {
2         require(_withdrawLockupEpochs >= _rewardLockupEpochs &&
  _withdrawLockupEpochs <= 56, "_withdrawLockupEpochs: out of range");
  // <= 2 week
3         withdrawLockupEpochs = _withdrawLockupEpochs;
4         rewardLockupEpochs = _rewardLockupEpochs;
5
```

DESCRIPTION	It is possible for the operator to set the *withdrawLockupEpochs* and *rewardLockupEpochs* to 0, withdraw their rewards, then change it back
RECOMMENDATION	Add a lower bound to prevent this. A timelock can also resolve this issue.
RESOLUTION	**Deployed Contracts:** Open **Rev-5 Contracts:** Closed The project team has implemented the recommendation.

No Limits For Discount rate

FINDING ID	#0013
SEVERITY	Informational
STATUS	Open
LOCATION	Treasury.sol -> 381-387

```
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	**Deployed Contracts:** Open **Rev-5 Contracts:** Closed The project team has implemented the recommendation. **Project team comment:** “All variables and code is altered to accommodate future forking and comments added for clarity”

Comments Do Not Match Contract Code

FINDING ID	#0014
SEVERITY	Informational
STATUS	Open
LOCATION	Acropolis.sol -> 138-139

```
1    withdrawLockupEpochs = 4; // Lock for 6 epochs (36h) before
  release withdraw
2    rewardLockupEpochs = 2; // Lock for 3 epochs (18h) before release
  claimReward    !!! THIS IS ALTERED TO SMALLER PERIODS
```

LOCATION	Treasury.sol -> 268

```
1        maxSupplyExpansionPercent = 600; // Upto 4.0% supply for
  expansion
```

LOCATION	Treasury.sol -> 278-280

```
1        // First 28 epochs with 4.5% expansion
2        bootstrapEpochs = 14;
3        bootstrapSupplyExpansionPercent = 600;
```

LOCATION	Treasury.sol -> 356-358

```
1        // DAO FUND - 12%
2        // DEVS WALLET - 3%
3        // TEAM WALLET - 5%
```

LOCATION	Treasury.sol -> 366-371

```
        require(_daoFund != address(0), "zero");
        require(_daoFundSharedPercent <= 1500, "out of range"); // <= 15%
        require(_devFund != address(0), "zero");
        require(_devFundSharedPercent <= 350, "out of range"); // <= 3.5%
        require(_teamFund != address(0), "zero");
        require(_teamFundSharedPercent <= 550, "out of range"); // <= 5.5%
```

DESCRIPTION	The first three code blocks have comments which do not reflect the code. The values in the fourth and fifth blocks do not match each other. There is a discrepancy with what is noted in the project documentation as well. https://docs.basedfinance.io/meeting-the-team/team-allocations.
RECOMMENDATION	Make sure that the comments reflect the code and the documentation.
RESOLUTION	**Deployed Contracts:** Open **Rev-5 Contracts:** Closed The project team has implemented the recommendation. **Project team comment:** “All variables and code is altered to accommodate future forking and comments added for clarity”

Unnecessary Override

FINDING ID	#0015
SEVERITY	Informational
STATUS	Open
LOCATION	Based.sol -> 84-92

```
1     function transferFrom(
2         address sender,
3         address recipient,
4         uint256 amount
5     ) public override returns (bool) {
6         _transfer(sender, recipient, amount);
7         _approve(sender, _msgSender(), allowance(sender,
  _msgSender()).sub(amount, "ERC20: transfer amount exceeds
  allowance"));
8         return true;
9     }
```

DESCRIPTION	This function appears to be exactly the same as the function that it overrides.
RECOMMENDATION	Confirm that these functions are identical and remove the derived implementation.
RESOLUTION	**Project team comment:** “All variables and code is altered to accommodate future forking and comments added for clarity”

Unused Modifier

FINDING ID	#0016
SEVERITY	Informational
STATUS	Closed
LOCATION	• BShareSwapper.sol -> 48-52: *modifier isSwappable() {*

DESCRIPTION	The noted modifier is never used.
RECOMMENDATION	Remove the unused modifier or incorporate it into the contract functionality.
RESOLUTION	The project team has implemented the recommendation.

Confusing Token Name

FINDING ID	#0017
SEVERITY	Informational
STATUS	Open
LOCATION	BShareFtmZap.sol -> 23-24

```
1    // @NATIVE - native token that is not a part of our zap-in LP
2    address private NATIVE;
```

LOCATION	BShareFtmZap.sol -> 23-24

```
1    function zapInToken(address _from, uint256 amount, address _to,
  address routerAddr, address _recipient) external {
```

DESCRIPTION	The native token typically refers to the chain's token (eg. ETH, FTM, etc). Variables named *_from* and *_to* are typically used for transferring tokens between addresses, not the swap tokens.
RECOMMENDATION	Consider using an alternative name for the "native" token. Use another name for *_from* and *_to*, for instance, *_in* and *_out*.
RESOLUTION	**Project team comment:** “All variables and code is altered to accommodate future forking and comments added for clarity”

Repetitive Swap Functions Code

FINDING ID	#0018
SEVERITY	Informational
STATUS	Open
LOCATION	• BasedTombZap.sol -> 178-197: *function _swapTokenToLP(address _from, uint256 amount, address _to, address recipient, address routerAddr) private returns (uint256) {* • BasedTombZap.sol -> 199-207: *function _swapNativeToLP(address _LP, uint256 amount, address recipient, address routerAddress) private returns (uint256) {* • BasedTombZap.sol -> 210-225: *function _swapNativeToEqualTokensAndProvide(address token0, address token1, uint256 amount, address routerAddress, address recipient) private returns (uint256, uint256, uint256) {* • BasedTombZap.sol -> 227-244: *function _swapNativeForToken(address token, uint256 value, address recipient, address routerAddr) private returns (uint256) {* • BasedTombZap.sol -> 246-263: *function _swapTokenForNative(address token, uint256 amount, address recipient, address routerAddr) private returns (uint256) {* • BasedTombZap.sol -> 265-320: *function _swap(address _from, uint256 amount, address _to, address recipient, address routerAddr) private returns (uint256) {* • BasedTombZap.sol -> 322-377: *function _estimateSwap(address _from, uint256 amount, address _to, address routerAddr) private view returns (uint256) {* • BShareFtmZap.sol -> 180-199: *function _swapTokenToLP(address _from, uint256 amount, address _to, address recipient, address routerAddr) private returns (uint256) {* • BShareFtmZap.sol -> 201-214: *function _swapNativeToLP(address _LP, uint256 amount, address recipient, address routerAddress) private returns (uint256) {* • BShareFtmZap.sol -> 216-228: *function _swapHalfNativeAndProvide(address token, uint256 amount, address routerAddress, address recipient) private returns (uint256, uint256, uint256) {* • BShareFtmZap.sol -> 230-238: *function _swapNativeToEqualTokensAndProvide(address token0, address*

	token1, uint256 amount, address routerAddress, address recipient) private returns (uint256, uint256, uint256) { • BShareFtmZap.sol -> 240-257: *function _swapNativeForToken(address token, uint256 value, address recipient, address routerAddr) private returns (uint256) {* • BShareFtmZap.sol -> 259-276: *function _swapTokenForNative(address token, uint256 amount, address recipient, address routerAddr) private returns (uint256) {* • BShareFtmZap.sol -> 278-333: *function _swap(address _from, uint256 amount, address _to, address recipient, address routerAddr) private returns (uint256) {* • BShareFtmZap.sol -> 335-390: *function _estimateSwap(address _from, uint256 amount, address _to, address routerAddr) private view returns (uint256) {*

DESCRIPTION	The noted functions contain significant code duplication. This combined with a similar naming convention can cause the contracts to be difficult to interpret.
RECOMMENDATION	Refactor the functions to reduce the code duplications.
RESOLUTION	**Project team comment:** "All variables and code is altered to accommodate future forking and comments added for clarity"

Incorrect Arguments For _dustDistribution

FINDING ID	#0020
SEVERITY	Medium Risk
STATUS	Open
LOCATION	rev-2 - Zapper.sol -> 239-247

```
1    args._token = args._in == IUniswapV2Pair(args._out).token0() ?
  IUniswapV2Pair(args._out).token1() :
  IUniswapV2Pair(args._out).token0();
2    .....
3   _dustDistribution(args._amount.sub(args._swapAmt), args._otherAmt,
  pair._amountToken0, pair._amountToken1, args._in, args._token,
  args._recipient);
```

DESCRIPTION	This code assumes that *args._amount* corresponds to *Token0* from the LP pair, but the reality is that *args._amount* could be *Token0* or *Token1* as evidenced by the conditional state in the first line of the snippet. For example: if *pair._token* == *Token0* then the value of *args._amount* corresponds to *Token1* but will always be supplied as the *token0* parameter to *_dustDistribution*.
RECOMMENDATION	Identify which token *args._amount* corresponds to and pass in the arguments to *_dustDistribution* accordingly.
RESOLUTION	**Deployed Contracts:** Open **Rev-5 Contracts:** Closed The project team has implemented the recommendation. **Project team comment:** “Args work as intended for our lps and were tested prior”

IZapper Interface Does Not Match Zapper

FINDING ID	#0021
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	rev-2 - BShareSwapper.sol -> 79-85

```
1    function swap(address _in, uint256 amount, address out, address
  recipient, address routerAddr, uint256 slippage) private returns
  (uint256) {
2          try IZapper(zapper)._swap(_in, amount, out, recipient,
  routerAddr , slippage) returns (uint256 _bshareAmount) {
3              return uint256(_bshareAmount);
4          } catch {
5              revert("Treasury: failed to consult BSHARE price from the
  oracle");
6          }
7      }
```

LOCATION	rev-2 - IZapper.sol -> 6

```
1    function _swap(address _in, uint256 amount, address out, address
  recipient, address routerAddr, uint256 slippage) external returns
  (uint256);
```

LOCATION	rev-2 - Zapper.sol-> 359-415

```
1    function _swap(address _in, uint256 amount, address out, address
  recipient, address routerAddr, uint256 slippage) private returns
  (uint256) {
2          // ...
3      }
```

DESCRIPTION	The interface *IZapper.sol* declares the function *_swap* with the visibility *external*, but this function is actually declared as *private* on the *Zapper.sol* contract. Calling the *_swap* function from *BShareSwapper.sol* will fail.
RECOMMENDATION	Remove the function from the interface or update the

	function visibility.
RESOLUTION	The project team updated the function visibility.

Oracle May Drift

FINDING ID	#0022
SEVERITY	Informational
STATUS	Open
LOCATION	rev-2 - Oracle.sol -> 80-104

```
function update() external checkEpoch {
        (uint256 price0Cumulative, uint256 price1Cumulative, uint32 blockTimestamp) = UniswapV2OracleLibrary.currentCumulativePrices(address(pair));
        uint32 timeElapsed = blockTimestamp - blockTimestampLast; // overflow is desired

        // Ensure that at least one full period has passed since the last update
        require(timeElapsed >= PERIOD, "UniswapPairOracle: PERIOD_NOT_ELAPSED");

        if (timeElapsed == 0) {
            // prevent divided by zero
            return;
        }

        // overflow is desired, casting never truncates
        // cumulative price is in (uq112x112 price * seconds) units so we simply wrap it after division by time elapsed
        unchecked {
            price0Average = FixedPoint.uq112x112(uint224((price0Cumulative - price0CumulativeLast) / timeElapsed));
            price1Average = FixedPoint.uq112x112(uint224((price1Cumulative - price1CumulativeLast) / timeElapsed));
        }

        price0CumulativeLast = price0Cumulative;
        price1CumulativeLast = price1Cumulative;
        blockTimestampLast = blockTimestamp;

        emit Updated(price0Cumulative, price1Cumulative);
    }
```

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	**Project team comment:** “We are talking to chainlink to use their oracles and all price feeds, if we don’t get them in a short amount of time - will redeploy contracts with overflow fixed”

DoS Can Occur With Unexpected Revert

FINDING ID	#0023
SEVERITY	High Risk
STATUS	Open
LOCATION	rev-3 - ProfitDistribution.sol -> 221-249

```
    function issueInterestToken(uint256 _rewardId) public onlyOperator
{
        RewardInfo storage reward = rewardInfo[_rewardId];
        require(reward.isActive, "No rewards");

        for (uint256 i = 0; i < stakers.length; ++ i) {
            address recipient = stakers[i];
            UserInfo storage user = userInfo[recipient];
            uint256 poolShare = getPoolShare(recipient);
            uint256 rewards = poolShare * reward.rewardsPerEpoch /
(1e18);

            // distribute income proportionally to their staked amount.

            if(rewards > 0) {

                //update pendingRewards
                user.pendingRewards[_rewardId] += rewards;
                emit PendingRewardIncrease(recipient,_rewardId,
rewards);

                //update totalRewards
                reward.totalRewards -= rewards;
                emit RewardDecrease(_rewardId, rewards);
            }

        }

        if (reward.totalRewards == 0 || reward.totalRewards <
reward.rewardsPerEpoch) {
            reward.isActive = false;
        }
    }
```

LOCATION	rev-3 - ProfitDistribution.sol -> 157-192

```
    function stakeTokens(uint256 _amount) external {
        address _sender = msg.sender;
        UserInfo storage user = userInfo[_sender];

        require(_amount > 0, "can't stake 0");

        // 1% fee calculation
        uint256 feeAmount = _amount * depositFee / 100000;
        uint256 depositAmount = _amount - feeAmount;

        //update totalBurned
        totalBurned += totalBurned;

        // Update the staking balance in map
        user.balance += depositAmount;
        emit UserStakedIncrease(_sender, depositAmount);

        //update TotalStaked
        totalStaked += depositAmount;
        emit TotalStakedIncrease(depositAmount);

        // Add user to stakers array if they haven't staked already
        if(!user.hasStaked) {
            stakers.push(_sender);
        }

        // Update staking status to track
        user.isStaking = true;
        user.hasStaked = true;

        // Transfer based tokens to contract for staking
        depositToken.safeTransferFrom(_sender, address(this), _amount);

        // burn based
        depositToken.safeTransfer(burnAddress, feeAmount);
    }
```

DESCRIPTION	The *issueInterestToken()* can revert if too many users are added to the array. The *issueInterestToken()* currently loops through each user in an array when distributing tokens. When a user stakes their tokens via `stakeTokens()` funcion, the user is added to the array. The network has a gas limit per block that can not be exceeded. A malicious user could use this by making puppet accounts to increase the loop size and make the *issueInterestToken()* function unusable.

RECOMMENDATION	We recommend redesigning the contract that doesn't require looping through each user. An example of such a contract would be a Masterchef contract which could be easily modified to distribute tokens in batches.
RESOLUTION	The contract has not been deployed yet. **Project team comment:** “Added safety measures, the contract is not deployed yet, all issues are taken care of.”

_swap Does Not Approve Or Transfer Tokens

FINDING ID	#0024
SEVERITY	Low Risk
STATUS	Closed
LOCATION	rev-3 - Zapper.sol -> 359-410

```
1    function _swap(address _in, uint256 amount, address out, address
  recipient, address routerAddr, uint256 slippage) private returns
  (uint256) {
2        IUniswapV2Router router = IUniswapV2Router(routerAddr);
3
4        address fromBridge = tokenBridgeForRouter[_in][routerAddr];
5        address toBridge = tokenBridgeForRouter[out][routerAddr];
6
7        address[] memory path;
8
9        // ...
10
11       uint256 tokenAmountEst = _estimateSwap(_in, amount, out,
  routerAddr);
12
13       uint256[] memory amounts =
  router.swapExactTokensForTokens(amount,
  tokenAmountEst.sub(tokenAmountEst.mul(slippage).div(10000)), path,
  recipient, block.timestamp);
14       return amounts[amounts.length - 1];
15   }
```

LOCATION	rev-3 - BShareSwapper.sol -> 79-85

```
1    function swap(address _in, uint256 amount, address out, address
  recipient, address routerAddr, uint256 slippage) private returns
  (uint256) {
2        try IZapper(zapper)._swap(_in, amount, out, recipient,
  routerAddr , slippage) returns (uint256 _bshareAmount) {
3            return uint256(_bshareAmount);
4        } catch {
5            revert("Treasury: failed to consult BSHARE price from the
  oracle");
6        }
7    }
```

DESCRIPTION	The function *swap*, which has visibility *public*, neither approves the *router* nor transfers tokens from the caller.

	This means that *_swap* will fail unless approval has previously been given to the router (via another function) and tokens have been transferred to the contract right before calling the function. Currently, the BShareSwapper.sol contract uses the *_swap* function incorrectly and will revert. Note This function may be more suited to use as an *internal* function.
RECOMMENDATION	Ensure the function is used appropriately by callers. Make the function *internal* or update it to work correctly without requiring other functions to be first called.
RESOLUTION	The project team made the function private.

Burn Address Can Be Updated

FINDING ID	#0025
SEVERITY	Low Risk
STATUS	Open
LOCATION	rev-3 - ProfitDistribution.sol -> 92-95

```
1     function updateBurnAddress(address _burnAddress) external
  onlyOperator {
2         burnAddress = _burnAddress;
3         emit UpdateBurnAddress(_burnAddress);
4     }
```

DESCRIPTION	There should be no reason to update the burnAddress. For example, if a malicious actor gets access to the Operator address they could change the burn address and receive those tokens instead.
RECOMMENDATION	Remove the function.
RESOLUTION	**Project team comment:** “Have a whitelist of dead address and treasury address, depends on how we decide to distribute(burn) fees.”

Updating Native Token Breaks Swap Paths

FINDING ID	#0026
SEVERITY	Low Risk
STATUS	Mitigated
LOCATION	rev-3 - Zapper.sol -> 535-541

```
1    function setUseNativeRouter(address router) external onlyOwner {
2        useNativeRouter[router] = true;
3    }
4
5    function removeNativeRouter(address router) external onlyOwner {
6        useNativeRouter[router] = false;
7    }
```

DESCRIPTION	Updating the *NATIVE* token address will invalidate any path that uses the previous address. This could result in many invalid paths or paths using low liquidity pools. Additionally, because slippage and price impact aren't implemented correctly yet, a malicious actor with access to the owner address could swap the native token out as a frontrunning measure to make someone use a malicious pool. (This can be prevented if a timelock is used or slippage/price impact is correctly implemented.) A malicious actor with permission to update *tokenBridgeForRouter* could update the route to a low liquidity pool allowing them to steal tokens. This is only possible as slippage and price impact checks inside the transaction currently. For this reason, permission to update *tokenBridgeForRouter* should be restricted to a timelock.
RECOMMENDATION	Remove the *setNativeToken()* function.
RESOLUTION	The owner is renounced on-chain.

Fixed Swap Path Lengths

FINDING ID	#0027
SEVERITY	Informational
STATUS	Mitigated
LOCATION	• rev-3 - Zapper.sol -> 317-334: *function _swapNativeForToken(address token, uint256 amount, address recipient, address routerAddr, uint256 slippage) private returns (uint256) {* • rev-3 - Zapper.sol -> 337-355: *function _swapTokenForNative(address token, uint256 amount, address recipient, address routerAddr, uint256 slippage) private returns (uint256) {* • rev-3 - Zapper.sol -> 359-415: *function _swap(address _in, uint256 amount, address out, address recipient, address routerAddr, uint256 slippage) private returns (uint256) {* • rev-3 - Zapper.sol -> 419-474: *function _estimateSwap(address _in, uint256 amount, address out, address routerAddr) private view returns (uint256) {*

DESCRIPTION	The current pathing design does not allow for much flexibility and is restricted to a limited number of steps in each path. This may prevent some tokens from being supported or cause some tokens to be routed along sub-optimal paths.
RECOMMENDATION	Allow paths of arbitrary length.
RESOLUTION	The owner is renounced on-chain.

Zapper Might Use Malicious Liquidity Pools

FINDING ID	#0028
SEVERITY	Informational
STATUS	Mitigated
LOCATION	• rev-3 - Zapper.sol -> 317-334: *function _swapNativeForToken(address token, uint256 amount, address recipient, address routerAddr, uint256 slippage) private returns (uint256) {* • rev-3 - Zapper.sol -> 337-355: *function _swapTokenForNative(address token, uint256 amount, address recipient, address routerAddr, uint256 slippage) private returns (uint256) {* • rev-3 - Zapper.sol -> 359-415: *function _swap(address _in, uint256 amount, address out, address recipient, address routerAddr, uint256 slippage) private returns (uint256) {* • rev-3 - Zapper.sol -> 419-474: *function _estimateSwap(address _in, uint256 amount, address out, address routerAddr) private view returns (uint256) {*

DESCRIPTION	When swapping tokens without an explicitly defined path, the zapper contract will convert to native liquidity in an attempt to find a path. This may result in swaps being routed through malicious liquidity pools. The likelihood of this happening will depend on how the UI displays it to the user.
RECOMMENDATION	Store an explicit path for each token and reject any tokens without an explicit path.
RESOLUTION	The owner is renounced on-chain.

Router Trust

FINDING ID	#0030
SEVERITY	Informational
STATUS	Open
LOCATION	rev-5 - Zapper.sol

DESCRIPTION	The *Zapper* contract relies on the return values of calls to the supplied *routerAddr* to verify that the swap has resulted in at least the minimum amount of tokens received (as specified by the caller). Should a malicious router be called, the return values could be faked and the user would receive less than the specified minimum. Note that the router still needs to be whitelisted and called by the user.
RECOMMENDATION	Ensure that all routers are fully vetted and trusted or check that the amount of tokens received from the router (the change in balance) is at least the minimum amount specified by that caller.
RESOLUTION	N/A

Static Analysis

Division Before Multiplication 1

FINDING ID	#0019
SEVERITY	Informational
STATUS	Open
LOCATION	distribution/BasedGenesisRewardPool.sol -> 161-172

```
1      // View function to see pending BASED on frontend.
2      function pendingBASED(uint256 _pid, address _user) external view
  returns (uint256) {
3          PoolInfo storage pool = poolInfo[_pid];
4          UserInfo storage user = userInfo[_pid][_user];
5          uint256 accBasedPerShare = pool.accBasedPerShare;
6          uint256 tokenSupply = pool.token.balanceOf(address(this));
7          if (block.timestamp > pool.lastRewardTime && tokenSupply !=
  0) {
8              uint256 _generatedReward =
  getGeneratedReward(pool.lastRewardTime, block.timestamp);
9              uint256 _basedReward =
  _generatedReward.mul(pool.allocPoint).div(totalAllocPoint);
10             accBasedPerShare =
  accBasedPerShare.add(_basedReward.mul(1e18).div(tokenSupply));
11         }
12         return
  user.amount.mul(accBasedPerShare).div(1e18).sub(user.rewardDebt);
13     }
```

LOCATION	distribution/BasedGenesisRewardPool.sol -> 197-201

```
1          if (totalAllocPoint > 0) {
2              uint256 _generatedReward =
  getGeneratedReward(pool.lastRewardTime, block.timestamp);
3              uint256 _basedReward =
  _generatedReward.mul(pool.allocPoint).div(totalAllocPoint);
4              pool.accBasedPerShare =
  pool.accBasedPerShare.add(_basedReward.mul(1e18).div(tokenSupply));
5          }
```

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	N/A

Division Before Multiplication 2

FINDING ID	#0029
SEVERITY	Low Risk
STATUS	Open
LOCATION	Rev-3 - ProfitDistribution.sol -> 221-249

```
1     function issueInterestToken(uint256 _rewardId) public onlyOperator
  {
2         RewardInfo storage reward = rewardInfo[_rewardId];
3         require(reward.isActive, "No rewards");
4
5         for (uint256 i = 0; i < stakers.length; ++ i) {
6             address recipient = stakers[i];
7             UserInfo storage user = userInfo[recipient];
8             uint256 poolShare = getPoolShare(recipient);
9             uint256 rewards = poolShare * reward.rewardsPerEpoch /
  (1e18);
10
11            // distribute income proportionally to their staked amount.
12
13            if(rewards > 0) {
14
15                //update pendingRewards
16                user.pendingRewards[_rewardId] += rewards;
17                emit PendingRewardIncrease(recipient,_rewardId,
  rewards);
18
19                //update totalRewards
20                reward.totalRewards -= rewards;
21                emit RewardDecrease(_rewardId, rewards);
22            }
23
24        }
25
26        if (reward.totalRewards == 0 || reward.totalRewards <
  reward.rewardsPerEpoch) {
27            reward.isActive = false;
28        }
29    }
```

DESCRIPTION	The share of the pool belonging to a single address is calculated with *getPoolShare()* and then this value is used to calculate the rewards received by that address. Because *getPoolShare()* divides the user's balance by the total staked amount a rounding error occurs when this value is used to calculate the rewards. This will cause a small number of tokens to be left behind in the contract.

RECOMMENDATION	Perform all multiplication first, then perform all division.
RESOLUTION	N/A

On-Chain Analysis

No Timelock

FINDING ID	#0031
SEVERITY	High Risk
STATUS	Open
LOCATION	BShareRewardPool 0xAc0fa95058616D7539b6Eecb6418A68e7c18A746 Oracle 0x8AA7e17f96647E08a54880521b2f3e7F99aa11ca Treasury 0x25430E2ACb1255453135b2c64436C6b1Ff8153C5

DESCRIPTION	The *operator* role of the following contracts has not been transferred to a timelock yet: • BShareRewardPool • Oracle • Treasury The *owner* role of the following contracts has not been transferred to a timelock yet: • Oracle
RECOMMENDATION	Transfer the *operator* and *owner* roles to a timelock contract. Obelisk recommends a minimum delay of 72 hrs.
RESOLUTION	**Project team comment:** "Will add after protocol takes shape and works as intended."

Multisig Low Amount Of Signers

FINDING ID	#0032
SEVERITY	Informational
STATUS	Open
LOCATION	BasedGenesisRewardPool.daoFundAddress 0xA0e0F462d66De459711BC721cE1fdCC3D9405831 BShareRewardPool.daoFundAddress 0xA0e0F462d66De459711BC721cE1fdCC3D9405831 Treasury.daoFund 0xA0e0F462d66De459711BC721cE1fdCC3D9405831

DESCRIPTION	The Multisig requires 2/10 signers to execute a transaction. This is a low proportion of the people with access. Owners: 0x084D4ba0a80bD5D35A0Eb6c47B417fc60cb351d0 0x386b5ab400a0546dB9C8831E5B0E6c1A77bE3885 0x387dfB7Fca2606AcaacD769Fb59803F6539C8269 0x39bB82e080E8a1b0895ab74b20a737498c56c18e 0x6Fd0C2F6ceF3B83c2a463D561739d6F7cA9CF809 0x85Ff34cCDD3cB36A4553D57009371be35eb72870 0x8B70966b3990bBb08f9D0287e568Bdac55b4F83B 0xC0Ad1703d2062Fb32db92Cb31492bd41220D16D1 0xF1b86a431f6c990fB37D57866Bdc7ebD2F151E72 0xa4fE067c4646c7b7cA8944C60490fDB176e3Acd3
RECOMMENDATION	Clearly identify each signer for reference. Require that a majority of the owners sign any transactions.
RESOLUTION	**Project team comment:** “Amount of signers will be bigger after treasury acquires steady income from fees and other protocols.”

External Addresses

Externally Owned Accounts

Multisig Owners

ACCOUNT	0x084D4ba0a80bD5D35A0Eb6c47B417fc60cb351d0 0x386b5ab400a0546dB9C8831E5B0E6c1A77bE3885 0x387dfB7Fca2606AcaacD769Fb59803F6539C8269 0x39bB82e080E8a1b0895ab74b20a737498c56c18e 0x6Fd0C2F6ceF3B83c2a463D561739d6F7cA9CF809 0x85Ff34cCDD3cB36A4553D57009371be35eb72870 0x8B70966b3990bBb08f9D0287e568Bdac55b4F83B 0xC0Ad1703d2062Fb32db92Cb31492bd41220D16D1 0xF1b86a431f6c990fB37D57866Bdc7ebD2F151E72 0xa4fE067c4646c7b7cA8944C60490fDB176e3Acd3
USAGE	0xA0e0F462d66De459711BC721cE1fdCC3D9405831 *BasedGenesisRewardPool.daoFundAddress* - Variable, no setter 0xA0e0F462d66De459711BC721cE1fdCC3D9405831 *BShareRewardPool.daoFundAddress* - Variable, no setter 0xA0e0F462d66De459711BC721cE1fdCC3D9405831 *Treasury.daoFund* - Variable
IMPACT	• receives elevated permissions as owner, operator, or other

Owner / Operator

ACCOUNT	0x1252E3f03E0caa840cbb35442d817a1686A62586
USAGE	0xA0e0F462d66De459711BC721cE1fdCC3D9405831 *BShareRewardPool.operator* - Variable 0x8AA7e17f96647E08a54880521b2f3e7F99aa11ca *Oracle.operator* - Variable *Oracle.owner* - Variable 0x25430E2ACb1255453135b2c64436C6b1Ff8153C5 *Treasury.operator* - Variable

IMPACT	• receives elevated permissions as owner, operator, or other

Dev Fund

ACCOUNT	0x0819855c478B7561Aa0df8412F0195c5Ee56ee02
USAGE	0x25430E2ACb1255453135b2c64436C6b1Ff8153C5 *Treasury.devFund* - Variable
IMPACT	• receives transfer of tokens deposited or minted by project

Team Fund

ACCOUNT	0x59bdD7397e3f988Af72fFA3245B1F5BB452aAa42
USAGE	0x25430E2ACb1255453135b2c64436C6b1Ff8153C5 *Treasury.teamFund* - Variable
IMPACT	• receives transfer of tokens deposited or minted by project

External Contracts

These contracts are not part of the audit scope.

Tomb

ADDRESS	0x6c021Ae822BEa943b2E66552bDe1D2696a53fbB7
USAGE	0x9Ec66B9409d4cD8D4a4C90950Ff0fd26bB39ad84 *BasedGenesisRewardPool.poolInfo(0).token* - Variable, no setter 0xa3C92bd0Fb52e0536FD49Eaa988B1b0C1C5a75b6 *BasedTombZap.NativeToken* - Variable
IMPACT	• ERC20 Token

WETH

ADDRESS	0x74b23882a30290451A17c44f4F05243b6b58C76d
USAGE	0x9Ec66B9409d4cD8D4a4C90950Ff0fd26bB39ad84 *BasedGenesisRewardPool.poolInfo(1).token* - Variable, no setter
IMPACT	• ERC20 Token

WFTM

ADDRESS	0x21be370D5312f44cB42ce377BC9b8a0cEF1A4C83
USAGE	0x9Ec66B9409d4cD8D4a4C90950Ff0fd26bB39ad84 *BasedGenesisRewardPool.poolInfo(2).token* - Variable, no setter 0xb4A3f001f86Cd68459e015026e344096A90d8620 *BShareFtmZap.NativeToken* - Variable
IMPACT	• ERC20 Token

USDC

ADDRESS	0x04068DA6C83AFCFA0e13ba15A6696662335D5B75
USAGE	0x9Ec66B9409d4cD8D4a4C90950Ff0fd26bB39ad84

	BasedGenesisRewardPool.poolInfo(3).token - Variable, no setter
IMPACT	• ERC20 Token

USDC

ADDRESS	0x04068DA6C83AFCFA0e13ba15A6696662335D5B75
USAGE	0x9Ec66B9409d4cD8D4a4C90950Ff0fd26bB39ad84 *BasedGenesisRewardPool.poolInfo(3).token* - Variable, no setter
IMPACT	• ERC20 Token

Tomb-Based Spooky LP

ADDRESS	0xaB2ddCBB346327bBDF97120b0dD5eE172a9c8f9E
USAGE	0x8AA7e17f96647E08a54880521b2f3e7F99aa11ca *Oracle.pair* - Variable, no setter 0x9Ec66B9409d4cD8D4a4C90950Ff0fd26bB39ad84 *BasedGenesisRewardPool.poolInfo(4).token* - Variable, no setter 0xAc0fa95058616D7539b6Eecb6418A68e7c18A746 BShareRewardPool.*poolInfo(0).token* - Variable, no setter
IMPACT	• ERC20 Token

BShare-FTM Spooky LP

ADDRESS	0x6F607443DC307DCBe570D0ecFf79d65838630B56
USAGE	0xAc0fa95058616D7539b6Eecb6418A68e7c18A746 BShareRewardPool.*poolInfo(1).token* - Variable, no setter
IMPACT	• ERC20 Token

BShare-Based Spooky LP

ADDRESS	0x5748b5Dd1f59342f85d170c48C427959c2f9f262
USAGE	0xAc0fa95058616D7539b6Eecb6418A68e7c18A746

	BShareRewardPool.*poolInfo(5).token* - Variable, no setter
IMPACT	• ERC20 Token

Curve.fi g3CRV RewardGauge Deposit A

ADDRESS	0xd4F94D0aaa640BBb72b5EEc2D85F6D114D81a88E
USAGE	0xAc0fa95058616D7539b6Eecb6418A68e7c18A746 BShareRewardPool.*poolInfo(3).token* - Variable, no setter
IMPACT	• ERC20 Token

Curve.fi g3CRV RewardGauge Deposit B

ADDRESS	0x00702BbDEaD24C40647f235F15971dB0867F6bdB
USAGE	0xAc0fa95058616D7539b6Eecb6418A68e7c18A746 BShareRewardPool.*poolInfo(4).token* - Variable, no setter
IMPACT	• ERC20 Token

BASED-miMATIC TOMB-V2-LP

ADDRESS	0x00702BbDEaD24C40647f235F15971dB0867F6bdB
USAGE	0xAc0fa95058616D7539b6Eecb6418A68e7c18A746 BShareRewardPool.*poolInfo(6).token* - Variable, no setter
IMPACT	• ERC20 Token

Appendix A - Reviewed Documents

Deployed Contracts

Document	Address
distribution/BasedGenesisRewardPool.sol	0x9Ec66B9409d4cD8D4a4C90950Ff0fd26bB39ad84
distribution/BShareRewardPool.sol	0xAc0fa95058616D7539b6Eecb6418A68e7c18A746
Acropolis.sol	0xE5009Dd5912a68B0D7C6F874cD0b4492C9F0e5cD
Based.sol	0x8D7d3409881b51466B483B11Ea1B8A03cdEd89ae
BasedTombZap.sol	0xa3C92bd0Fb52e0536FD49Eaa988B1b0C1C5a75b6
BBond.sol	0xC078285F16665B3F4bCe74AbDCF0f4C877de3E9f
BShare.sol	0x49C290Ff692149A4E16611c694fdED42C954ab7a
BShareFtmZap.sol	0xb4A3f001f86Cd68459e015026e344096A90d8620
BShareSwapper.sol	N/A
Distributor.sol	N/A
Oracle.sol	0x8AA7e17f96647E08a54880521b2f3e7F99aa11ca
ProfitDistribution.sol	N/A
SimpleERCFund.sol	N/A
Stater.sol	0x5F511b4B68D83b72dc687B56D845257368Cc7243
Timelock.sol	N/A
Treasury.sol	0x25430E2ACb1255453135b2c64436C6b1Ff8153C5

Libraries and Interfaces

interfaces/IAcropolis.sol interfaces/IBasisAsset.sol interfaces/IBShareRewardPool.sol interfaces/IDecimals.sol interfaces/IDistributor.sol interfaces/IERC20.sol interfaces/IHyperswapRouter01.sol

interfaces/IOracle.sol interfaces/IShare.sol interfaces/ISimpleERCFund.sol interfaces/ITaxable.sol interfaces/ITreasury.sol interfaces/IUniswapV2Callee.sol interfaces/IUniswapV2ERC20.sol interfaces/IUniswapV2Factory.sol interfaces/IUniswapV2Pair.sol interfaces/IUniswapV2Router.sol interfaces/IUniswapV2Router01.sol interfaces/IVault.sol interfaces/IWrappedFtm.sol lib/Babylonian.sol lib/FixedPoint.sol lib/Safe112.sol lib/SafeMath8.sol lib/TransferHelper.sol lib/UniswapV2Library.sol lib/UniswapV2OracleLibrary.sol lib/UQ112x112.sol owner/Operator.sol utils/ContractGuard.sol utils/Epoch.sol

Revisions

Revision 1	08e2ddbef63bc030ff8656fbf51d39d3da197c95
Revision 2	8b5dbd07214134ae54ead9835ffde7c255ca2d54
Revision 3	774e837b42b5e60cd9442050a6ce48021b7ea59f
Revision 4	73890113f8e6a2eeee26ae5cc6c445bd9f2d52d3
Revision 5	05af3b81109764b797e8d0cd8d3d26691ff1881c

Imported Contracts

OpenZeppelin	4.4.2

Appendix B - Risk Ratings

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

Appendix C - Finding Statuses

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

Appendix D - Audit Procedure

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

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.

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

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

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

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

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

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

Follow Obelisk Auditing for the Latest Information

 ObeliskOrg

 ObeliskOrg

OBELISK

Part of Tibereum Group