OBELISK

Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 86	2021-10-23	Zapmore, Donut, DoD4uFN	Audit Final

Audit Notes

Audit Date	2021-08-24 - 2021-10-22
Auditor/Auditors	Donut, DoD4uFN
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	OB585672569

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.

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

Project Name	Summit
Description	Summit Defi is bringing new and unique features to the #DeFi space, starting off with what we are calling "Yield Multiplying" launching first on $FTM
Website	https://twitter.com/SummitDefi
Contact	@architect_dev
Contact information	@architect_dev on TG
Token Name(s)	See Appendix A
Token Short	See Appendix A
Contract(s)	See Appendix A
Code Language	Solidity
Chain	FTM

Audit of *Summit*

> **Summit commissioned Obelisk at such a time in their development that Obelisk could give feedback on findings and the Summit team could swiftly solve issues found during continued development. All severe findings were either Closed or Mitigated.**

Obelisk was commissioned by Summit on the 17th of August 2021 to conduct a comprehensive audit of Summits' contracts. The following audit was conducted between the 24th of August 2021 and the 22nd of October 2021. 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 reason for the amount of time the audit took was because the Summit team worked on the contracts in tandem with the audit. Because of the workflow provided, Obelisk could continuously give Summit information on findings and recommended solutions to them. The Summit team always responded fast to any recommendation and made all the changes necessary to solve the issues found. The only left issue to comment on is that some functions are under a 24hour timelock instead of the 72hours that Obelisk recommends (see issue #41 for a comment).

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 Summit project.

Please read the full document for a complete understanding of the audit.

Summary Table

Audited Part	ID	Severity	Status
Approval Not Needed For Transfer	#0001	High Risk	Closed
Checking Balance To Ensure There Are Enough Funds To Launch Expedition	#0002	Medium Risk	Closed
Finished Pools Restart On Round Rollover	#0003	Medium Risk	Closed
Off-Chain Randomness Cannot Be Guaranteed	#0004	Medium Risk	Open
No Limit For Protocol Values	#0005	Medium Risk	Closed
Passthrough Rewards Are Sent To A Potential EOA Instead Of Expedition	#0006	Medium Risk	Mitigated
Faulty Assumption Of Zero Fees	#0007	Low Risk	Mitigated
Delegates Not Transferred	#0008	Low Risk	Closed
Redundant Mint Functions	#0009	Low Risk	Closed
Check For Balance Before Transfering	#0010	Low Risk	Closed
Pool Rewards Not Updated On Changes	#0011	Low Risk	Closed
Unbound Loop	#0012	Low Risk	Closed
Cartographer Enabled Token Allocations Can Be Out Of Sync	#0013	Low Risk	Closed
Referral Reward Amounts Can Exceed Available Rewards	#0014	Low Risk	Closed
Division Before Multiplication	#0015	Low Risk	Closed
Low Time Delay For Changing Passthrough	#0016	Low Risk	Closed
Total Emissions Calculation Includes Allocation Of Disabled Elevations	#0017	Low Risk	Closed

No List Of Allocated Tokens	#0018	Low Risk	Closed
Modified Copy Of Compound Timelock	#0019	Informational	Mitigated
Potential Parameter Mismatch With Vault	#0020	Informational	Closed
Potential Transfer Of 0 Tokens (Gas Optimization)	#0021	Informational	Closed
Referral Cycles Not Prevented	#0022	Informational	Partially Closed
Expedition isLive Always Returns True	#0023	Informational	Closed
Redundant Check-In Require	#0024	Informational	Closed
Elevation Unified Deposit Can Switch Totem	#0025	Informational	Closed
Unnecessary Multiplication Then Division (Gas Optimization)	#0026	Informational	Closed
Passthrough Strategy Can Be Retired With No Delay	#0027	Low Risk	Closed
Contract Values Can Be Constant Or Immutable (Gas Optimization)	#0028	Informational	Closed
Missing Zero Checks	#0029	Informational	Closed
No Events Emitted For Changes To Protocol Values	#0030	Informational	Closed
Unused Modifiers	#0031	Informational	Closed
Potential Underflows	#0032	Low Risk	Closed
Elevating With LP Can Result In Incorrect Amounts	#0033	Medium Risk	Closed
Elevation Rewards Are Combined Between Pools	#0034	Low Risk	Closed
No Check For Summit In SummitLP	#0035	Low Risk	Closed
Unable To Update Active Pools When Maximum Number Is Reached	#0036	Low Risk	Closed

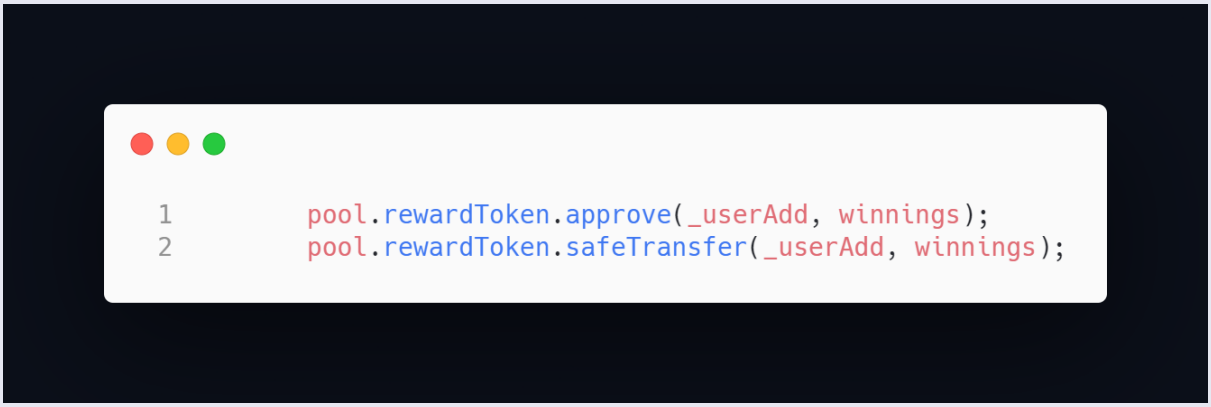
Extended Expeditions Miscalculate Reward Rate	#0037	Low Risk	Closed
Oasis Can Generate Rewards From Before Pool Launch	#0038	Low Risk	Closed
Maximum Start Round Offset Does Not Match Error Message	#0039	Informational	Closed
Unrestricted Harvest Function Allows Compounding To Any Pool	#0040	High Risk	Closed
Timelock Delay Is Short	#0041	Medium Risk	Partially Mitigated
Timelock Admin Not Correctly Initialized	#0042	Medium Risk	Closed
Cartographer Expedition Address Is An EOA	#0043	Medium Risk	Open
Vaults With Withdrawal Fees Deployed	#0044	Low Risk	Open

Findings

Manual Analysis

Approval Not Needed For Transfer

FINDING ID	#0001
SEVERITY	High Risk
STATUS	**Closed**
LOCATION	CartographerExpedition.sol -> 616-617



```
1        pool.rewardToken.approve(_userAdd, winnings);
2        pool.rewardToken.safeTransfer(_userAdd, winnings);
```

DESCRIPTION	The expedition provides approval and a safe transfer for a user's winnings. Because of the approval, a user will be able to double their winnings by withdrawing their approved limit from the expedition.
RECOMMENDATION	Remove the token approval.
MITIGATED/COMMENT	The token approval was removed. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Checking Balance To Ensure There Are Enough Funds To Launch Expedition

FINDING ID	#0002
SEVERITY	Medium Risk
STATUS	**Closed**
LOCATION	CartographerExpedition.sol -> 309-333

```
1     function restartExpeditionPool(uint16 _pid, uint256 _rewardAmount, uint256 _rounds)
2         public onlyOwner
3         poolExists(_pid)
4     {
5         ExpeditionPoolInfo storage pool = expeditionPoolInfo[_pid];
6         require (!pool.launched, "Expedition already running");
7         require(pool.rewardToken.balanceOf(address(this)) >= _rewardAmount, "Must have funds to
  cover expedition");
8
9         // Set state variables of restarted expedition
10        pool.launched = false;
11        pool.live = true;
12        pool.totalRoundsCount = _rounds;
13        pool.totalRewardAmount = _rewardAmount;
14        pool.roundEmission = _rewardAmount.div(_rounds);
15
16        // Expedition will be restarted for the following round
17        pool.startRound = elevationHelper.nextRound(EXPEDITION);
18
19        emit ExpeditionRestarted(_pid, address(pool.rewardToken), pool.totalRewardAmount,
  pool.totalRoundsCount);
20    }
```

DESCRIPTION	*addExpedition* doesn't distinguish between tokens newly added to re-launch the expeditions, and tokens held as rewards from the previous launch. As a result, the new launch can distribute users' existing rewards and then attempt to redistribute them for the next launch. This can cause users to be unable to withdraw their rewards.
RECOMMENDATION	Track the reward to be distributed and subtract that from the balance when extending or relaunching expeditions.
MITIGATED/COMMENT	The *rewardsMarkedForDist* variable was added to the *ExpeditionPoolInfo* struct. This will keep track of rewards that are reserved for distribution to users. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Finished Pools Restart On Round Rollover

FINDING ID	#0003
SEVERITY	Medium Risk
STATUS	**Closed**
LOCATION	CartographerExpedition.sol -> 502-505

```
} else if (nextRound >= expeditionPoolInfo[expeditionPID].startRound) {
    // Mark expedition as launched and running
    expeditionPoolInfo[expeditionPID].launched = true;
}
```

DESCRIPTION	Pools are launched automatically if the round number is higher than their start round. Expedition pools are marked as not launched on the round they expire, but in the next round, they will be marked as launched again. As a result, the pool will advance a round and emit new rewards even though it is terminated. This will allocate more rewards than the expedition contract has in its balance.
RECOMMENDATION	Ensure that pools are not automatically re-launched during the rollover process.
MITIGATED/COMMENT	Pools are changed to only launch on their start round. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Off-Chain Randomness Cannot Be Guaranteed

FINDING ID	#0004
SEVERITY	Medium Risk
STATUS	**Open**
LOCATION	ElevationHelper.sol -> 295-308

```
1    function receiveSealedSeed(bytes32 _sealedSeed)
2        public
3        onlyTrustedSeeder
4    {
5        require(nextSeedRoundAvailable(), "Already sealed seeded");
6
7        // Increment seed round and set next seed round end timestamp
8        seedRound += 1;
9        seedRoundEndTimestamp += (baseRoundDuration * seedRoundDurationMult);
10
11       // Store new sealed seed for next round of round rollovers
12       sealedSeed[seedRound] = _sealedSeed;
13       futureBlockNumber[seedRound] = block.number + 1;
14   }
```

DESCRIPTION	Randomness generated off-chain cannot be guaranteed as truly random. A malicious actor in control of the protocol may manipulate the operation of the trusted seeder. Additionally, if the seed is not generated in the 60s timeframe, the seed will effectively be null.
RECOMMENDATION	Use a cryptographically verifiable randomness oracle.
MITIGATED/COMMENT	Project Team Comment: “On-Chain VRF is not available on the Fantom Opera EVM. Elevation Farms and Expeditions rely heavily on randomness, so we have opted into the tried and tested "trustedSeeder" randomness flow. Additionally, Summit DeFi does not feature randomness that would award a select few users obscenely large rewards (such as a lottery), instead, large groups of users are winning or losing together. This reduces the effective value of attempting to abuse our randomness solution. The Summit team is committed to fairness, and we will be monitoring our randomness solution closely until

	On-Chain VRF is released on the Fantom chain. Our randomness solution is open source, and we are working to make it available for other projects on FTM to hook into and use”

No Limit For Protocol Values

FINDING ID	#0005
SEVERITY	Medium Risk
STATUS	**Closed**
LOCATION	• Cartographer.sol -> 267-274: function setTotalSummitPerSecond(uint256 _amount) public onlyOwner • Cartographer.sol -> 279-288: function setSummitDistributionProfile(uint256 _staking, uint256 _dev) public onlyOwner • Cartographer.sol -> 446-464 : function setTokenSharedAlloc(IBEP20 _token, uint256 _allocation) public onlyOwner tokenAllocExists(_token) • CartographerOasis.sol -> 195-212: function add(uint16 _pid, uint8, bool _live, IBEP20 _token, uint16 _feeBP) external override onlyCartographer nonDuplicated(_token) • CartographerOasis.sol -> 223-238: function set(uint16 _pid, bool _live, uint16 _feeBP) external override onlyCartographer poolExists(_pid) • CartographerElevation.sol -> 376-400: function add(uint16 _pid, uint8 _elevation, bool _live, IBEP20 _token, uint16 _feeBP) external override onlyCartographer nonDuplicated(_token, _elevation) • CartographerElevation.sol -> 411-426: function set(uint16 _pid, bool _live, uint16 _feeBP) external override onlyCartographer poolExists(_pid)

DESCRIPTION	The following values can be set arbitrarily high, potentially breaking the functionality of the contracts: • CartographerOasis.pool.feeBP • CartographerExpedition.pool.feeBP • Cartographer.summitPerSecond • Cartographer.referralsSummitPerSecond • Cartographer.devSummitPerSecond • Cartographer.totalSharedAlloc
RECOMMENDATION	Add an upper limit to the values.
MITIGATED/COMMENT	*Cartographer.summitPerSecond* was limited to 1e18 (1 ether) per second. The dev share of the emissions was limited to 25% of the total emissions.

	The fees of the oasis and expedition contracts are limited in the contract to 4%. Reviewed in commit [7a89c5840bec5c74aa503709c796346ff75684ba]

Passthrough Rewards Are Sent To A Potential EOA Instead Of Expedition

FINDING ID	#0006
SEVERITY	Medium Risk
STATUS	**Mitigated**
LOCATION	Cartographer.sol -> 87 Cartographer.sol -> 93 Cartographer.sol -> 251-255

```
address public expedAdd;
```

```
ISubCart cartographerExpedition;
```

```
function setExpedAdd(address _expedAdd) public {
    require(msg.sender == expedAdd, "Forbidden");
    expedAdd = _expedAdd;
    emit SetExpedAdd(msg.sender, _expedAdd);
}
```

DESCRIPTION	Profits from the passthrough rewards are expected to be

	sent to the *cartographerExpedition*. However, there is a separate *expedAdd* to which the rewards are sent. This can result in profits being redirected towards an externally owned address.
RECOMMENDATION	Remove the ability to set the *expedAdd* and direct all profits to the *cartographerExpedition* address. Alternatively, add a timelock to ensure that users are able to react to changes to the address.
MITIGATED/COMMENT	Added a 72-hour timelock to provide advanced notice of changes to the expedition address. Reviewed in commit [7a89c5840bec5c74aa503709c796346ff75684ba]

Faulty Assumption Of Zero Fees

FINDING ID	#0007
SEVERITY	Low Risk
STATUS	**Mitigated**
LOCATION	BeefyVaultPassthrough.sol -> 63-77

```
    function deposit(address, address)
        external override
        onlyCartographer
    {
        // Transfer funds from cartographer to this contract
        uint256 cartographerBalance = passthroughToken.balanceOf(cartographer);
        passthroughToken.safeTransferFrom(cartographer, address(this), cartographerBalance);

        // Deposit all funds in this contract into vault
        uint256 amount = passthroughToken.balanceOf(address(this));
        IBeefyVault(vault).depositAll();

        // Update running token value
        usersTokensInVault = usersTokensInVault.add(amount);
    }
```

LOCATION	MasterChefPassthrough.sol -> 84-98

```
1     function deposit(address _expedAdd, address _devAdd)
2         external override
3         onlyCartographer
4     {
5         uint256 cartographerBalance =
  passthroughToken.balanceOf(cartographer);
6         passthroughToken.safeTransferFrom(cartographer,
  address(this), cartographerBalance);
7
8         uint256 amount =
  passthroughToken.balanceOf(address(this));
9
10        IMasterChef(masterChef).deposit(masterChefPid,
  amount);
11
12        usersTokensInVault =
  usersTokensInVault.add(amount);
13
14        distributeRewards(_expedAdd, _devAdd);
15    }
```

LOCATION	MasterChefPassthrough.sol -> 105-121

```
    function withdraw(uint256 _amount, address _expedAdd, address _devAdd)
        external override
        onlyCartographer
        returns (uint256)
    {
        IMasterChef(masterChef).withdraw(masterChefPid, _amount);

        // Return withdrawn amount back to cartographer
        passthroughToken.safeTransfer(cartographer, _amount);

        // Distribute the remaining rewards in this contract
        distributeRewards(_expedAdd, _devAdd);

        usersTokensInVault = usersTokensInVault.sub(_amount);

        return _amount;
    }
```

DESCRIPTION	BeefyVaultPassthrough: Updating the variable *usersTokensInVault* assumes that the tokens being staked are equal to the tokens sent, which may not be the case. Beefy vaults usually have a withdrawal fee, but in some cases, there is a deposit fee instead. (Note: sometimes the fees come from the underlying farming contract, not the vault) MasterChefPassthrough: There are MasterChef contracts with deposit or withdrawal fees. In order to make sure the functionality works in every case a safer way to track *usersTokensInVault* is needed. General: Tokens with transaction fees may result in the same issues.
RECOMMENDATION	BeefyVaultPassthrough: Use *balance*() to get the current

	balance, deposit the funds, then use balance *balance*() again. This allows the contract to calculate the difference between the initial and final amounts. Instead of storing the number of *want* tokens, track the number of shares instead. MasterChefPassthrough: Make use of *userInfo(pid, address)* to determine how many tokens were staked/withdrawn.
MITIGATED/COMMENT	A check was added to ensure that MasterChefPassthrough does not use underlying contracts with fees. Additionally, the transferred amounts are checked in order to avoid the assumption of no deposit fees on the underlying contracts. Tokens with built-in transfer fees will still be affected. The user stake is recorded in the sub-cartographer contracts become out of sync. Any passthrough strategy with withdrawal fees can also cause the recorded stake to become out of sync if retired. Project Team Comment: "The plan is only to use non-reflective tokens, on masterchefs / vaults with no deposit fee. This accounts for 95% of the passthrough strategies we are aiming to use." Reviewed in commit [bb1a005902d0df7d75b0584a5682812d27d6a564]

Delegates Not Transferred

FINDING ID	#0008
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	BEP20.sol -> 113-116 BEP20.sol -> 149-157 SummitToken.sol -> 8

```
1     function transfer(address recipient, uint256 amount)
  public override returns (bool) {
2         _transfer(_msgSender(), recipient, amount);
3         return true;
4     }
```

```
1     function transferFrom (address sender, address
  recipient, uint256 amount) public override returns (bool) {
2         _transfer(sender, recipient, amount);
3         _approve(
4             sender,
5             _msgSender(),
6             _allowances[sender][_msgSender()].sub(amount,
  'BEP20: transfer amount exceeds allowance')
7         );
8         return true;
9     }
```

DESCRIPTION	Delegates are not transferred when tokens are transferred.
RECOMMENDATION	Use *_moveDelegates* to ensure that delegates are properly accounted for when SUMMIT tokens are transferred.
MITIGATED/COMMENT	The delegate system was removed from SummitToken. Reviewed in commit [7a89c5840bec5c74aa503709c796346ff75684ba]

Redundant Mint Functions

FINDING ID	#0009
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	BEP20.sol -> 203-206 SummitToken.sol -> 11-16

```
1     function mint(uint256 amount) public onlyOwner returns
  (bool) {
2         _mint(_msgSender(), amount);
3         return true;
4     }
```

```
1     function mint(address _to, uint256 _amount) public
  onlyOwner {
2         _mint(_to, _amount);
3         _moveDelegates(address(0), _delegates[_to],
  _amount);
4     }
5
6     function mintInitial(address _to, uint256 _amount)
  public onlyOwner { mint(_to, _amount); }
```

DESCRIPTION	3 variations of mint are available to the SummitToken's owner. The function *mintInitial* can be called multiple times though it appears that it is meant to create the initial supply.
RECOMMENDATION	Remove the redundant mint implementations and ensure

	that tokens cannot be freely minted.
MITIGATED/COMMENT	The *SummitToken* mint functions were replaced with a single *mintTo* function which is used by the cartographer to mint rewards. Note: *BEP20.mint* is never used. Reviewed in commit [7a89c5840bec5c74aa503709c796346ff75684ba]

Check For Balance Before Transfering

FINDING ID	#0010
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	CartographerExpedition.sol -> 609-618

```
    function harvestWinnings(ExpeditionPoolInfo storage pool, UserInfo storage user, address _userAdd)
        internal
    {
        // Get calculated harvestable winnings
        uint256 winnings = harvestableWinnings(pool, user, _userAdd);

        // Transfer winnings to user
        pool.rewardToken.approve(_userAdd, winnings);
        pool.rewardToken.safeTransfer(_userAdd, winnings);
    }
```

DESCRIPTION	*harvestWinnings* doesn't check the available balance before transferring the *winnings*. That could result in Cartographer's *switchTotem*, *deposit*, *withdraw,* and *elevate* unusably.
RECOMMENDATION	Considering adding a check of the current balance in order to limit the maximum transferred amount.
MITIGATED/COMMENT	A check was to pool management function to ensure that sufficient rewards are available to users. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Pool Rewards Not Updated On Changes

FINDING ID	#00011
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	CartographerOasis.sol -> 223-238

```
1     function set(uint16 _pid, bool _live, uint16 _feeBP)
2         external override
3         onlyCartographer poolExists(_pid)
4     {
5         OasisPoolInfo storage pool = oasisPoolInfo[_pid];
6
7         // If live status of pool changes, update
  cartographer allocations
8         if (pool.live != _live) {
9             if (_live) {
  cartographer.enableTokenAtElevation(pool.token, OASIS); }
10            else {
  cartographer.disableTokenAtElevation(pool.token, OASIS); }
11        }
12
13        // Update internal pool states
14        pool.live = _live;
15        pool.feeBP = _feeBP;
16    }
```

LOCATION	CartographerElevation.sol -> 411-426

```
1     function set(uint16 _pid, bool _live, uint16 _feeBP)
2         external override
3         onlyCartographer poolExists(_pid)
4     {
5         ElevationPoolInfo storage pool =
  elevationPoolInfo[_pid];
6
7         // If live status changes, update cartographer
  allocations
8         if (pool.live != _live) {
9             if (_live) {
  cartographer.enableTokenAtElevation(pool.token,
  pool.elevation); }
10            else {
  cartographer.disableTokenAtElevation(pool.token,
  pool.elevation); }
11        }
12
13        // Update internal pool states
14        pool.live = _live;
15        pool.feeBP = _feeBP;
16    }
```

DESCRIPTION	Changing the value of *pool.live* will not update the pool. If the pool is turned off, rewards since the last time the pool was updated will be lost. If the pool is turned on, rewards may be generated from the last time the pool was updated.
RECOMMENDATION	Update the pool whenever making changes to the live variable.
MITIGATED/COMMENT	The pool is updated before any changes in the set functions noted above. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Unbound Loop

FINDING ID	#0012
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	Looping over CartographerElevation.elevationPIDs.length: • CartographerElevation.sol -> 434-436 Looping over CartographerElevation.poolsAtElev.length • CartographerElevation.sol -> 638-653 Looping over CartographerElevation.harvestableWinnings().mostRecentCompletedRoundIndex: • CartographerElevation.sol -> 803-805 Looping over CartographerElevation.poolsAtElevation[_elevation].length • CartographerElevation.sol -> 955-960 • CartographerElevation.sol -> 981-988 Looping over CartographerExpedition.expeditionPIDs.length • CartographerExpedition.sol -> 494-506 • CartographerExpedition.sol -> 656-661 Looping over CartographerExpedition.harvestableWinnings().currRound: • CartographerExpedition.sol -> 589-591

DESCRIPTION	Unbound loops may revert due to the gas fee limit.
RECOMMENDATION	Add start and end parameters to iterate over a range of expeditions or rounds.
MITIGATED/COMMENT	The rollover process was separated into two steps: • *Cartographer.rollover()* handles updating the ElevationHelper • *Cartographer.rolloverPools()* handles updating the elevation and expedition contracts This separation allows the contract functionality to avoid unbound loops when rolling over pools. The *totemRunningWinningsMult* mechanism was added to

both the Elevation and Expedition to prevent the iteration over a potentially unlimited number of rounds.

A limit of 12 staked pools per user per elevation was added. This prevents unbound looping when switching totems.

Reviewed in commit [bb1a005902d0df7d75b0584a5682812d27d6a564]

Cartographer Enabled Token Allocations Can Be Out Of Sync

FINDING ID	#0013
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	CartographerOasis.sol -> 207 CartographerElevation.sol -> 389

```
1            live: _live,
2
```

LOCATION	CartographerOasis.sol -> 230-233

```
1        if (pool.live != _live) {
2            if (_live) { cartographer.enableTokenAtElevation(pool.token, OASIS); }
3            else { cartographer.disableTokenAtElevation(pool.token, OASIS); }
4        }
```

LOCATION	CartographerOasis.sol -> 177

```
1         cartographer.enableTokenAtElevation(_token, OASIS);
2
```

LOCATION	CartographerElevation.sol -> 418-421

```
1         if (pool.live != _live) {
2             if (_live) {
  cartographer.enableTokenAtElevation(pool.token,
  pool.elevation); }
3             else {
  cartographer.disableTokenAtElevation(pool.token,
  pool.elevation); }
4         }
```

LOCATION	CartographerElevation.sol -> 651

```
1
   cartographer.enableTokenAtElevation(elevationPoolInfo[pidA
  tElev].token, elevationPoolInfo[pidAtElev].elevation);
2
```

DESCRIPTION	Cartographer and sub-cartographers track the overall allocations of tokens using a combination of *pool.live*, *pool.launched*, and whether the token is enabled in the

	cartographer. However, there are some instances where these are not necessarily updated in sync. For example, in *CartographerOasis*, a pool can be created with *live* initially set to false. However, it will automatically enable the token when registering the pool. Therefore, setting *live* to true via *CartographerOasis.set()* will then modify the allocated values a second time. In *CartographerElevation*, the token is automatically enabled when the pool is launched. A similar issue occurs if the pool was initially created with *live* set to false.
RECOMMENDATION	Ensure that pools tokens are enabled and disabled in the Cartographer while taking both the live and launched status into account.
MITIGATED/COMMENT	Oasis and Elevation were modified to ensure that the Cartographer values are correctly synchronized. Reviewed in commit [7a89c5840bec5c74aa503709c796346ff75684ba]

Referral Reward Amounts Can Exceed Available Rewards

FINDING ID	#0014
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	Cartographer.sol -> 1077-1085 SummitReferrals.sol -> 46-53

```
1     function redeemRewards(address _userAdd, uint256
  _amount) external onlySubCartographer {
2         // Transfers rewards to user
3         safeSummitTransfer(_userAdd, _amount);
4
5         // If the user has been referred, add the 1% bonus
  to that user and their referrer
6
   summitReferrals.addReferralRewardsIfNecessary(_userAdd,
  _amount);
7
8         emit RedeemRewards(_userAdd, _amount);
9     }
```

```
1     function addReferralRewardsIfNecessary(address referee,
  uint256 amount) external onlyCartographer {
2         if (referrerOf[referee] == address(0)) { return; }
3         uint256 additionalReward = amount.div(100);
4         pendingReferralRewards[roundIndex]
  [referrerOf[referee]] += additionalReward;
5         pendingReferralRewards[roundIndex][referee] +=
  additionalReward;
6         totalReferralRewards[referrerOf[referee]] +=
  additionalReward;
7         totalReferralRewards[referee] += additionalReward;
8     }
```

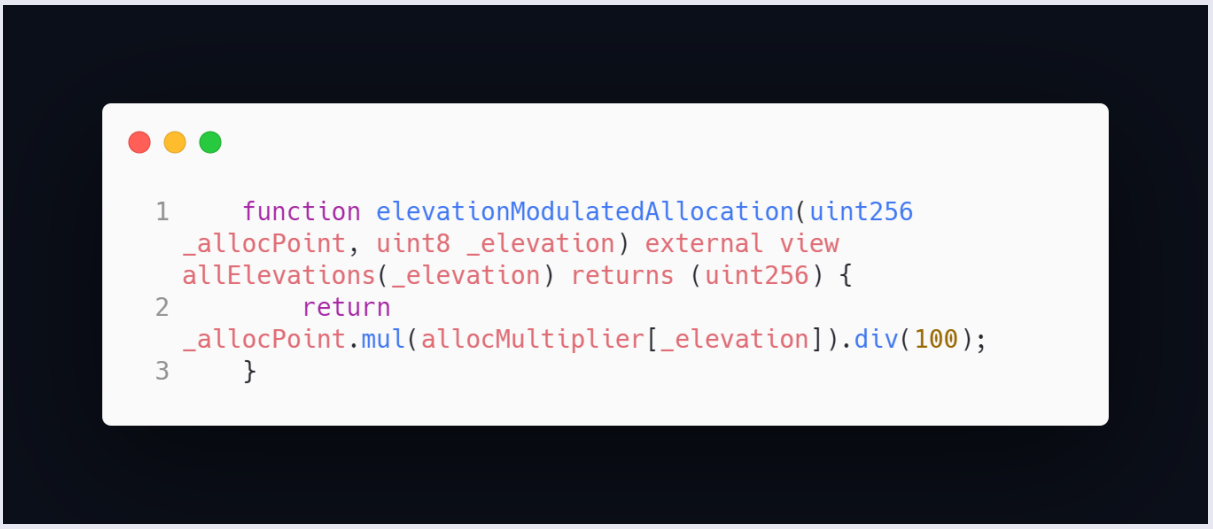
DESCRIPTION	The function *Cartographer.redeemRewards* adds 1% of all harvested winnings for a referrer and referee. However, this 1% counted for all harvested winnings regardless of the time frame or rounds from which they were earned. The *SummitReferrals* contract burns unclaimed Summit tokens at the end of every referral round. As a result, referral rewards can be added to users from past referral rounds. This will cause referral reward amounts to exceed the available reward balance.
RECOMMENDATION	Ensure that the referral reward balance matches what is available to be claimed.
MITIGATED/COMMENT	An 'escape hatch' mechanism was added to guarantee that referral reward amounts are always available. This may allow referral rewards from past rounds. Reviewed in commit [7a89c5840bec5c74aa503709c796346ff75684ba]

Division Before Multiplication

FINDING ID	#0015
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	Cartographer.sol -> 803-808 Cartographer.sol -> 813-816 ElevationHelper.sol -> 206-208

```
1     function elevationModulatedAllocation(uint16 _pid)
  public view returns (uint256) {
2         // Escape early if the pool is not live
3         if (!isLive(_pid)) { return 0; }
4         // Fetch the modulated base allocation for the
  token at elevation
5         return
  elevationHelper.elevationModulatedAllocation(tokenBaseAlloc
  [token(_pid)], poolElevation[_pid]);
6     }
```

```
1     function tokenElevationShares(IBEP20 _token, uint8
  _elevation) internal view returns (uint256) {
2         return stakedSupply(tokenElevationPid[_token]
  [_elevation])
3
  .mul(elevationModulatedAllocation(tokenElevationPid[_token]
  [_elevation]));
4     }
```



```
1     function elevationModulatedAllocation(uint256
  _allocPoint, uint8 _elevation) external view
  allElevations(_elevation) returns (uint256) {
2         return
  _allocPoint.mul(allocMultiplier[_elevation]).div(100);
3     }
```

DESCRIPTION	The function *ElevationHelper.elevationModulatedAllocation()* scales allocation points by a predetermined amount per elevation. Because it returns the result divided by the denominator, rounding errors may occur especially for small allocations. For example, an allocation of *3* at the *FIVETHOUSAND* elevation would be multiplied by 125 then divided by 100 then rounded down to *3* instead of the expected *3.75*.
RECOMMENDATION	Ensure divisions are performed after all multiplications are complete to avoid rounding errors.
MITIGATED/COMMENT	The calculation was modified to remove the division entirely. Reviewed in commit [7a89c5840bec5c74aa503709c796346ff75684ba]

Low Time Delay For Changing Passthrough

FINDING ID	#0016
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	Cartographer.sol -> 569

```
1
2
```

```
delayUnlock: block.timestamp + 24 hours
```

DESCRIPTION	The time delay for changing the passthrough strategy of a token is 24 hours. A timelock of at least 72 hours is expected.
RECOMMENDATION	Change the time delay to 72 hours.
MITIGATED/COMMENT	The time delay was set to 72 hours. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Total Emissions Calculation Includes Allocation Of Disabled Elevations

FINDING ID	#0017
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	Cartographer.sol -> 825-842

```
1     function tokenElevationEmissionMultiplier(IBEP20
_token, uint8 _elevation)
2         public view
3         returns (uint256)
4     {
5         // Shares for all elevation are summed. For each
elevation the shares are calculated by:
6         //   . The staked supply of the pool at elevation
multiplied by
7         //   . The modulated allocation of the pool at
elevation
8         uint256 totalTokenShares =
tokenElevationShares(_token, OASIS)
9             .add(tokenElevationShares(_token, TWOTHOUSAND))
10            .add(tokenElevationShares(_token,
FIVETHOUSAND))
11            .add(tokenElevationShares(_token,
TENTHOUSAND));
12
13        // Escape early if nothing is staked in any of the
token's pools
14        if (totalTokenShares == 0) { return 0; }
15
16        // Divide the target pool (token + elevation)
shares by total shares (as calculated above)
17        return tokenElevationShares(_token,
_elevation).mul(1e12).div(totalTokenShares);
18    }
```

DESCRIPTION	The calculation of emissions is calculated using the total deposits of all pools of a given token, whether they are live or launched or otherwise. This will dilute the earnings.
RECOMMENDATION	Only use active pools when calculating emissions.

MITIGATED/COMMENT	A new *isEarning()* function was added to the sub-cartographer contracts. This is used to get the total tokens of active pools. Reviewed in commit [7a89c5840bec5c74aa503709c796346ff75684ba]

No List Of Allocated Tokens

FINDING ID	#0018
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	Cartographer.sol -> 112

```
1     mapping(IBEP20 => bool) public tokenAllocExistence;
                  // Whether an allocation has been created
  for a specific token
2
```

DESCRIPTION	It is difficult to track what tokens have any allocation.
RECOMMENDATION	Add an array that lists all tokens with allocation which is synchronized to *tokenAllocExistence*.
MITIGATED/COMMENT	An array listing allocated tokens was added. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Modified Copy Of Compound Timelock

FINDING ID	#0019
SEVERITY	Informational
STATUS	**Mitigated**
LOCATION	Timelock.sol -> 65-76

```
1     function setPendingAdmin(address pendingAdmin_) public
  {
2         // allows one time setting of admin for deployment
  purposes
3         if (admin_initialized) {
4             require(msg.sender == address(this),
  "Timelock::setPendingAdmin: Call must come from
  Timelock.");
5         } else {
6             require(msg.sender == admin,
  "Timelock::setPendingAdmin: First call must come from
  admin.");
7             admin_initialized = true;
8         }
9         pendingAdmin = pendingAdmin_;
10
11        emit NewPendingAdmin(pendingAdmin);
12    }
```

DESCRIPTION	Timelock contract was modified in order to allow unrestricted control until it is activated.
RECOMMENDATION	No code modification is necessary. Ensure that timelock is initialized on-chain after setup.
MITIGATED/COMMENT	N/A

Potential Parameter Mismatch With Vault

FINDING ID	#0020
SEVERITY	Informational
STATUS	**Closed**
LOCATION	BeefyVaultPassthrough.sol -> 33-48 MasterChefPassthrough.sol -> 34-54

```
1     constructor(
2         address _cartographer,
3         address _vault,
4         IBEP20 _token
5     ) public {
6         require(_cartographer != address(0), "Cartographer
  missing");
7         require(_vault != address(0), "Vault missing");
8         require(address(_token) != address(0), "Passthrough
  token missing");
9
10        cartographer = _cartographer;
11        vault = _vault;
12        passthroughToken = _token;
13
14        passthroughToken.approve(vault, uint256(-1));
15        passthroughToken.approve(cartographer,
  uint256(-1));
16    }
```

```
    constructor(
        address _cartographer,
        address _masterChef,
        uint256 _masterChefPid,
        IBEP20 _token,
        IBEP20 _rewardToken
    ) public {
        require(_cartographer != address(0), "Cartographer missing");
        require(_masterChef != address(0), "MasterChef missing");
        require(address(_token) != address(0), "Passthrough token missing");
        require(address(_rewardToken) != address(0), "Reward token missing");

        cartographer = _cartographer;
        masterChef = _masterChef;
        masterChefPid = _masterChefPid;
        passthroughToken = _token;
        rewardToken = _rewardToken;

        passthroughToken.approve(masterChef, uint256(-1));
        passthroughToken.approve(cartographer, uint256(-1));
    }
```

DESCRIPTION	The *_token* parameter is referring to the token that the vault/pool is handling. Human mistakes are possible here, and there are no checks to prevent them.
RECOMMENDATION	BeefyVaultPassthrough: Use the vault's *want*() method. This returns an IRC20 address of this token. Alternatively use *strategy*().*want*() to get the address of the *_token* directly from the strategy contract. MasterChefPassthrough: Use the MasterChef's *poolInfo(pid)* to get the pool's info, including *lpToken*.
MITIGATED/COMMENT	A check was added to ensure that passthrough contracts will match their underlying vault/farm contracts. Reviewed in commit [7a89c5840bec5c74aa503709c796346ff75684ba]

Potential Transfer Of 0 Tokens (Gas Optimization)

FINDING ID	#0021
SEVERITY	Informational
STATUS	**Closed**
LOCATION	BeefyVaultPassthrough.sol -> 84-89 MasterChefPassthrough.sol -> 73-78

```
1     function distributeRemainingBalance(address _expedAdd,
  address _devAdd) internal {
2         approveDistributorsIfNecessary(_expedAdd, _devAdd);
3         uint256 toDistribute =
  passthroughToken.balanceOf(address(this));
4         passthroughToken.safeTransfer(_expedAdd,
  toDistribute.mul(92).div(100));
5         passthroughToken.safeTransfer(_devAdd,
  toDistribute.mul(8).div(100));
6     }
```

LOCATION	SummitReferrals.sol -> 60-65

```
1     function
  burnUnclaimedReferralRewardsAndRolloverRound(address
  referee) external onlyCartographer {
2         uint256 burnReward =
  summit.balanceOf(address(this)).div(100);
3         summit.transfer(referee, burnReward);
4         summit.transfer(burnAdd,
  summit.balanceOf(address(this)));
5         roundIndex += 1;
6     }
```

DESCRIPTION	If the amount of tokens being transferred is zero, this

	function will waste gas.
RECOMMENDATION	Add an if statement to check whether any tokens are being transferred.
MITIGATED/COMMENT	Transfer functions are now only called when there are tokens to transfer. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Referral Cycles Not Prevented

FINDING ID	#0022
SEVERITY	Informational
STATUS	**Partially Closed**
LOCATION	SummitReferrals.sol -> 34-40

```
1     function createReferral(address referrerAddress) public
  {
2         require(referrerAddress != msg.sender, "Cant refer
  yourself");
3         require(referrerOf[msg.sender] == address(0),
  "Already been referred");
4         require(referrerOf[referrerAddress] != msg.sender,
  "No reciprocal referrals");
5         referrerOf[msg.sender] = referrerAddress;
6         emit ReferralCreated(referrerAddress, msg.sender);
7     }
```

DESCRIPTION	Self-referrals and reciprocal referrals are prevented. However, three or more addresses may refer to each other in a cycle.
RECOMMENDATION	One way to address this would be to track all referrals and their sub-referrals to prevent cycles. However, this may be costly in terms of gas.
MITIGATED/COMMENT	Cycles of up to length 3 are prevented. Project Team Comment: "For referral cycles, without some deep looping to make sure that there aren't ever n-length cycles, there will always be cyclical referring available. I'll add a 3-cycle check, but getting to 4+ seems like diminishing returns." Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Expedition isLive Always Returns True

FINDING ID	#0023
SEVERITY	Informational
STATUS	**Closed**
LOCATION	CartographerExpedition.sol -> 241-243

```
1     function isLive(uint16) external view override returns
  (bool) {
2         return true;
3     }
```

DESCRIPTION	The *isLive*() function always returns true. However it can be toggled for a pool using the *disableExpeditionPool*() or *enableExpeditionPool*() functions. The *live* variable does not appear to impact any functionality except being able to extend an expedition pool.
RECOMMENDATION	Ensure that this is the intended behavior.
MITIGATED/COMMENT	*isLive()* was renamed to *isEarning* and now returns the status of the requested pool. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Redundant Check-In Require

FINDING ID	#0024
SEVERITY	Informational
STATUS	**Closed**
LOCATION	• CartographerExpedition.sol -> 848: require(_amount > 0 && user.staked > 0 && user.staked >= _amount, "Bad withdrawal"); • CartographerOasis.sol -> 471: require(_amount > 0 && user.staked > 0 && user.staked >= _amount, "Bad withdrawal"); • CartographerOasis.sol -> 492: require(_amount > 0 && user.staked > 0 && user.staked >= _amount, "Bad transfer")

DESCRIPTION	These functions check if *user.staked* > *0* while the other two conditions cover this requirement already.
RECOMMENDATION	Remove *user.staked* > 0 from *require* condition.
MITIGATED/COMMENT	Redundant checks were removed. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Elevation Unified Deposit Can Switch Totem

FINDING ID	#0025
SEVERITY	Informational
STATUS	**Closed**
LOCATION	CartographerElevation.sol -> 1161

```
1         userTotem[_userAdd][pool.elevation] = _totem;
2
```

DESCRIPTION	*CartographerElevation.unifiedDeposit()* can change the user's totem at a given elevation. This may result in the user's tokens being recorded as split across multiple totems. *CartographerElevation.deposit()* and *CartographerElevation.crossCompound()* check for the user's current totem and so guard against this. However, *CartographerElevation.elevateDeposit()* does not provide the same protection. A user's totem being out of sync with their totem balances can prevent them and other users from interacting with the contract successfully. This is safeguarded by the Cartographer contract itself.
RECOMMENDATION	No change is required.
MITIGATED/COMMENT	N/A

Unnecessary Multiplication Then Division (Gas Optimization)

FINDING ID	#0026
SEVERITY	Informational
STATUS	**Closed**
LOCATION	CartographerElevation.sol -> 336-340 CartographerElevation.sol -> 469

```
1     finalTotemRewards[i + 1] =
  pool.totemRoundRewards[i].add(
2         emissionToBringCurrent.mul(1e12)
3             .mul(pool.totemSupplies[i]).div(pool.supply)
4             .div(1e12)
5         );
```

```
1         for (uint8 i = 0; i <
  pool.totemRoundRewards.length; i++) {
2             pool.totemRoundRewards[i] =
  pool.totemRoundRewards[i].add(summitReward.mul(1e12).mul(po
  ol.totemSupplies[i]).div(pool.supply).div(1e12));
3         }
```

DESCRIPTION	The variables above are multiplied by *1e12* and then divided by *1e12*. This is typically done in order to keep enough precision during division operations. However, multiplying two token amounts (typically with 18 decimals) should provide sufficient precision.
RECOMMENDATION	Remove *.mul(1e12)* and *.div(1e12)* from the calculations.

MITIGATED/COMMENT	Unnecessary operations were removed. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Passthrough Strategy Can Be Retired With No Delay

FINDING ID	#0027
SEVERITY	Low Risk
STATUS	**Open**
LOCATION	Cartographer.sol -> 552-561

```
1    function retireTokenPassthroughStrategy(IBEP20 _token)
2         public
3         onlyOwner
4     {
5         require(tokenPassthroughStrategy[_token] !=
  address(0), "No passthrough strategy to retire");
6         address retiredTokenPassthroughStrategy =
  tokenPassthroughStrategy[_token];
7         internalRetireTokenPassthroughStrategy(_token);
8
9         emit PassthroughStrategyRetired(address(_token),
  retiredTokenPassthroughStrategy);
10    }
```

DESCRIPTION	The passthrough strategies can be retired with no time delay.
RECOMMENDATION	Add a timelock for retiring passthrough strategies.
MITIGATED/COMMENT	N/A

Elevating With LP Can Result In Incorrect Amounts

FINDING ID	#0033
SEVERITY	Medium Risk
STATUS	**Closed**
LOCATION	revised contract 0ff22344d8caed8e1e04ebbd9807ef9c30a5651f Cartographer.sol -> 1069-1099

```
        (uint256 elevateAmount, uint256 expedSummitLpElevateAmount) = subCartographer(sourceElev)
            .elevateWithdraw(
                _sourcePid,

                // If performing a SUMMIT LP elevate INTO an Expedition, the amount to withdraw is the amount of SUMMIT LP instead of the standard token
                isSummitLpElevate && targetElev == EXPEDITION ? _expedSummitLpAmount : _amount,

                // ONLY If performing a SUMMIT LP elevate FROM an Expedition, the amount to withdraw is the amount of SUMMIT LP
                isSummitLpElevate && sourceElev == EXPEDITION ? _expedSummitLpAmount : 0,

                msg.sender
            );


        // Elevating SUMMIT LP requires fiddling with how amounts are converted
        (elevateAmount, expedSummitLpElevateAmount) = subCartographer(targetElev)
            .elevateDeposit(
                _targetPid,

                // If performing a SUMMIT LP elevate
                    // If Elevating FROM an Expedition --> Amount to elevate is the amount of SUMMIT LP withdrawn from the Expedition Source Pool
                    // Else Not Elevating FROM, MUST be Elevating SUMMIT LP INTO an Expedition --> Amount of SUMMIT token to elevate is 0
                // Else it is a standard Elevate of any token OR a SUMMIT token elevate into or out of Expedition --> Amount of token to elevate is what was withdrawn from the source pool or expedition
                isSummitLpElevate ? (sourceElev == EXPEDITION ? expedSummitLpElevateAmount : 0) : elevateAmount,

                // Will only be NON ZERO if this is a SUMMIT LP elevate event INTO an Expedition
                (isSummitLpElevate && targetElev == EXPEDITION) ? elevateAmount : 0,

                _totem,
                msg.sender
            );
```

DESCRIPTION	Elevating LP tokens pass a modified value to the elevateWithdraw and elevateDeposit functions. This can lead to the unexpected movement of deposits between pools as well as the potential loss of deposits. Elevating LP between expeditions will result in _expedSummitLpAmount tokens and LP being transferred, not just tokens. Deposited SUMMIT tokens can be lost in the case of

	elevating from an elevation to the expedition contract while passing a non-zero _expedSummitLpAmount: • line 1074: _expedSummitLpAmount tokens will be withdrawn. • line 1092: 0 tokens will be deposited. • expedition will record that LP was elevated.
RECOMMENDATION	Ensure that the correct quantities of tokens are withdrawn and deposited.
MITIGATED/COMMENT	The elevate functions were re-factored to simplify their operation and resolve the finding. Reviewed in commit [bb1a005902d0df7d75b0584a5682812d27d6a564]

Elevation Rewards Are Combined Between Pools

FINDING ID	#0034
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	revised contract 86d4a3d3dc2e90d722ab0208d2a73d8b4dda998d CartographerElevation.sol -> 676-692

```
1         // Iterate through active pools of elevation, sum total rewards earned (all totems), and winning totems's rewards
2         uint256 elevTotalRewards = 0;
3         uint256 winningTotemRewards = 0;
4         for (uint16 activePoolSlot = 0; activePoolSlot < 24; activePoolSlot++) {
5             // Early exit if active pool slot is empty
6             if (elevActivePools[_elevation][activePoolSlot] == 0) continue;
7
8             // Bring pool current
9             updatePool(elevActivePools[_elevation][activePoolSlot]);
10
11            // Add round rewards of pool and winning totem to elevation round reward accumulators
12            elevTotalRewards += elevationPoolInfo[elevActivePools[_elevation][activePoolSlot]].roundRewards;
13            winningTotemRewards += elevationPoolInfo[elevActivePools[_elevation][activePoolSlot]].totemRoundRewards[winningTotem];
14        }
15
16        // Calculate the winnings multiplier of the round that just ended from the combined reward amounts
17        uint256 elevWinningsMult = winningTotemRewards == 0 ? 0 : elevTotalRewards.mul(1e12).div(winningTotemRewards);
```

DESCRIPTION	The proportion of rewards of each pool in the Elevation contract are flattened by dividing the sum of all rewards and the sum of all totem rewards. This will cause the incorrect calculation of winnings.

RECOMMENDATION	Revert to calculating reward multipliers on a per-pool basis.
MITIGATED/COMMENT	The project team has stated that this change in reward calculations was intentional. Project team comment: “User’s generated yield (in SUMMIT) over the duration of a round, this yield is contributed to each user's selected totem. The amount of yield generated by a farm is determined by the emission multiplier of that farm, and the user's yield is based on their staked amount. At the end of a round, the winning totem's yield multiplier (which will be passed down to all user's who have selected the winning totem) is calculated as so: winning totem's yield multiplier = total yield of all totems / winning totem's yield Previously, when a round ended, this calculation was done on each individual farm's yield held by each totem. However, at higher elevations like The Mesa and The Summit, the larger number of competing totems resulted in lower and less consistent yield held by each totem, and wildly variable Yield Multipliers for the winning totems. Because this multiplier is calculated for each farm individually, each farm will have a different Yield Multiplier, even though the winning totem is shared across all farms. With this update, when the round ends, the yield held by each totem is summed across all active pools at an elevation, and the summed values are used in the Yield Multiplier calculation above. The summed yields are less prone to variability and lead to fewer edge cases and a more consistent experience at higher elevations overall. Additionally, the winning totem's Yield Multiplier is now shared across all active farms of an elevation. Functionally, this means that two users that share a totem, and each contribute the same yield to that totem during a round, will win the same amount when that round ends, regardless of which farm they contributed yield from.”

No Check For Summit In SummitLP

FINDING ID	#0035
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	revised contract 86d4a3d3dc2e90d722ab0208d2a73d8b4dda998d Cartographer.sol -> 176-215

```
function initialize(
    SummitToken _summit,
    ILiquidityPair _summitLp,
    address _ElevationHelper,
    address _SummitReferrals,
    address _CartographerOasis,
    address _CartographerElevation,
    address _CartographerExpedition
)
    external
    initializer onlyOwner
{
    // ...
}
```

DESCRIPTION	The *intialize* function does not check that the *summitLP* is a liquidity pair with Summit as one of the tokens. The rest of the contracts assume that this is the case.
RECOMMENDATION	Add a check to ensure that the *summitLP* is the correct type of address.
MITIGATED/COMMENT	The project team has implemented the recommended fix. Reviewed in commit cd072c2a56c85674cc04e7f54bc2937442029ee2

Unable To Update Active Pools When Maximum Number Is Reached

FINDING ID	#0036
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	revised contract 86d4a3d3dc2e90d722ab0208d2a73d8b4dda998d CartographerElevation.sol -> 375-394

```
1     function updateActivePoolList(ElevationPoolInfo storage
  pool, bool _active) internal {
2         if (pool.active == _active) return;
3 
4         require(elevActivePoolsCount[pool.elevation] < 24,
  "Too many active pools");
5         pool.active = _active;
6         elevActivePoolsCount[pool.elevation] =
  elevActivePoolsCount[pool.elevation]
7             .add(_active ? 1 : 0)
8             .sub(!_active ? 1 : 0);
9 
10        uint16 targetPid = _active ? 0 : pool.pid;
11        uint16 replacementPid = _active ? pool.pid : 0;
12 
13        // Iterate through list of active pools and add pid
  to first available activePoolSlot
14        for (uint8 activePoolSlot = 0; activePoolSlot < 24;
  activePoolSlot++) {
15            if (elevActivePools[pool.elevation]
  [activePoolSlot] == targetPid) {
16                elevActivePools[pool.elevation]
  [activePoolSlot] = replacementPid;
17                return;
18            }
19        }
20    }
```

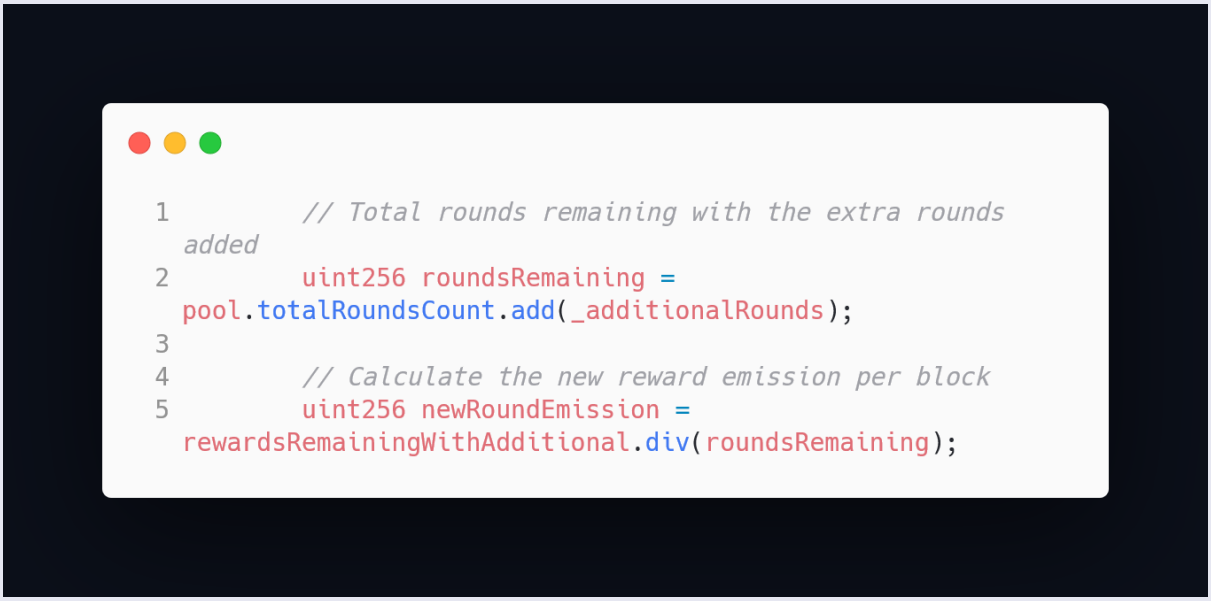
LOCATION	revised contract 86d4a3d3dc2e90d722ab0208d2a73d8b4dda998d CartographerExpedition.sol -> 349-368

```
    function updateActiveExpedsList(ExpeditionPoolInfo storage pool, bool _active) internal {
        if (pool.active == _active) return;

        require(activeExpedsCount < 24, "Too many active expeditions");
        pool.active = _active;
        activeExpedsCount = activeExpedsCount
            .add(_active ? 1 : 0)
            .sub(!_active ? 1 : 0);

        uint16 targetPid = _active ? 0 : pool.pid;
        uint16 replacementPid = _active ? pool.pid : 0;

        // Iterate through list of active pools and add pid to first available activePoolSlot
        for (uint8 activePoolSlot = 0; activePoolSlot < 24; activePoolSlot++) {
            if (activeExpeds[activePoolSlot] == targetPid) {
                activeExpeds[activePoolSlot] = replacementPid;
                return;
            }
        }
    }
```

DESCRIPTION	The *require* check in both update functions noted prevents any changes once the maximum number of pools has been reached.
RECOMMENDATION	Allow the removal of an active pool even at the maximum number of pools.
MITIGATED/COMMENT	The project team has implemented the recommended fix. Reviewed in commit cd072c2a56c85674cc04e7f54bc2937442029ee2

Extended Expeditions Miscalculate Reward Rate

FINDING ID	#0037
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	revised contract 86d4a3d3dc2e90d722ab0208d2a73d8b4dda998d CartographerExpedition.sol -> 476-480



```
1         // Total rounds remaining with the extra rounds
  added
2         uint256 roundsRemaining =
  pool.totalRoundsCount.add(_additionalRounds);
3
4         // Calculate the new reward emission per block
5         uint256 newRoundEmission =
  rewardsRemainingWithAdditional.div(roundsRemaining);
```

DESCRIPTION	The remaining rewards of an extended expedition pool are calculated as being distributed over all rounds instead of only the remaining rounds. This will result in lower-than-expected emissions.
RECOMMENDATION	Correctly calculate the remaining rounds to get the emissions per round.
MITIGATED/COMMENT	The number of rounds is now correctly calculated. Reviewed in commit 2111be7899f0d956f1af7391966af2440b6a0a8f

Oasis Can Generate Rewards From Before Pool Launch

FINDING ID	#0038
SEVERITY	Low Risk
STATUS	**Closed**
LOCATION	revised contract 86d4a3d3dc2e90d722ab0208d2a73d8b4dda998d CartographerOasis.sol -> 267-268

```
1         // Mint Summit according to pool allocation and
  token share in pool, retrieve amount of summit minted for
  staking
2         uint256 summitReward =
  cartographer.mintPoolSummit(pool.lastRewardTimestamp,
  pool.token, OASIS);
```

DESCRIPTION	The Oasis may generate rewards for staked tokens before it is launched. It uses the *pool.lastRewardTimestamp* to determine the duration of rewards to mint. A user could potentially deposit tokens when the contract is deployed, then once the oasis pools launch, they can receive rewards as if they had staked for a long time. Note that the *rewards()* view function has the same error.
RECOMMENDATION	Prevent the reward calculation from referring to times before the pool contracts are launched.
MITIGATED/COMMENT	The project team has implemented the recommended fix. Reviewed in commit 2111be7899f0d956f1af7391966af2440b6a0a8f

Maximum Start Round Offset Does Not Match Error Message

FINDING ID	#0039
SEVERITY	Informational
STATUS	**Closed**
LOCATION	revised contract 86d4a3d3dc2e90d722ab0208d2a73d8b4dda998d CartographerExpedition.sol -> 387 CartographerExpedition.sol -> 502

```
1         require(_startRoundOffset <= 7, "Must start within
  the week");
```

LOCATION	revised contract 86d4a3d3dc2e90d722ab0208d2a73d8b4dda998d ElevationHelper.sol -> 69-70

```
1     uint256 constant baseRoundDuration = 7200;
                  // Duration (seconds) of the smallest
  round chunk
2     uint256[5] public durationMult = [0, 2, 4, 8, 24];
                  // Number of round chunks for each
  elevation
```

DESCRIPTION	The maximum start round offset for expeditions is noted as 7 rounds or 1 week. The duration of a base round is 2 hours, therefore an expedition round will be 48 hours and 7 rounds will be two weeks.
RECOMMENDATION	Update the error message or maximum offset to match each other.

MITIGATED/COMMENT	Round durations were changed to be 24 hours. Reviewed in commit 2111be7899f0d956f1af7391966af2440b6a0a8f

Unrestricted Harvest Function Allows Compounding To Any Pool

FINDING ID	#0040
SEVERITY	High Risk
STATUS	**Closed**
LOCATION	revised contract cd072c2a56c85674cc04e7f54bc2937442029ee2 CartographerElevation.sol -> 1198-1232

```
    function harvestElevation(uint8 _elevation, uint16 _crossCompoundPid, address _userAdd)
        external override
        validUserAdd(_userAdd) elevationInteractionsAvailable(_elevation)
        returns (uint256)
    {
        // ...

        // Cross compound, else harvest to user
        if (harvestable > 0) {
            if (_crossCompoundPid != 0){

             unifiedDeposit(elevationPoolInfo[_crossCompoundPid], userInfo[_crossCompoundPid][_userAdd], harvestable, totem, _userAdd, true);
            } else {
                cartographer.redeemRewards(_userAdd, harvestable);
            }
        }

        return harvestable;
    }
```

LOCATION	revised contract cd072c2a56c85674cc04e7f54bc2937442029ee2 CartographerElevation.sol -> 1330-1332

```
1            // Only take deposit fee on standard deposit
2            if (!_isInternalTransfer)
3                amountAfterFee =
  depositTokenManagement(pool.pid, _amount, _userAdd);
```

DESCRIPTION	The *harvestElevation()* function is accessible to all users. It allows callers to specify a *_crossCompoundPid*. A compounding harvest will call *unifiedDeposit()* as an internal transfer, and record harvested summit tokens as being deposited as the pool's tokens. This will allow a user to drain another pool of staked tokens.
RECOMMENDATION	Ensure that the *harvestElevation()* can only be called by the cartographer.
MITIGATED/COMMENT	The function was restricted with the *onlyCartographer* modifier. Reviewed in commit 2111be7899f0d956f1af7391966af2440b6a0a8f

Static Analysis

Contract Values Can Be Constant Or Immutable (Gas Optimization)

FINDING ID	#0028
SEVERITY	Informational
STATUS	**Closed**
LOCATION	• ElevationHelper.sol -> 70: uint256 baseRoundDuration = 3600; • ElevationHelper.sol -> 82: uint256 referralDurationMult = 24; • SummitReferrals.sol -> 12: address burnAdd = 0x000000000000000000000000000000000000dEaD;

DESCRIPTION	Variables that do not change during the operation of a contract can be marked *constant* or *immutable* to reduce gas costs and improve code readability.
RECOMMENDATION	Mark these variables as *constant* or *immutable* as appropriate.
MITIGATED/COMMENT	Variables were marked as constant. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Missing Zero Checks

FINDING ID	#0029
SEVERITY	Informational
STATUS	**Closed**
LOCATION	• SummitReferrals.sol -> 19-22: function enable(address _trueSummit) external onlyCartographer • Cartographer.sol -> 242-246: function setDevAdd(address _devAdd) public • Cartographer.sol -> 251-255: function setExpedAdd(address _expedAdd) public • Cartographer.sol -> 259-263: function setTrustedSeederAdd(address _trustedSeederAdd) public onlyOwner

DESCRIPTION	Functions don't check for a zero address before assigning variables.
RECOMMENDATION	Add a check for zero address if deemed necessary.
MITIGATED/COMMENT	Zero address checks were added. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

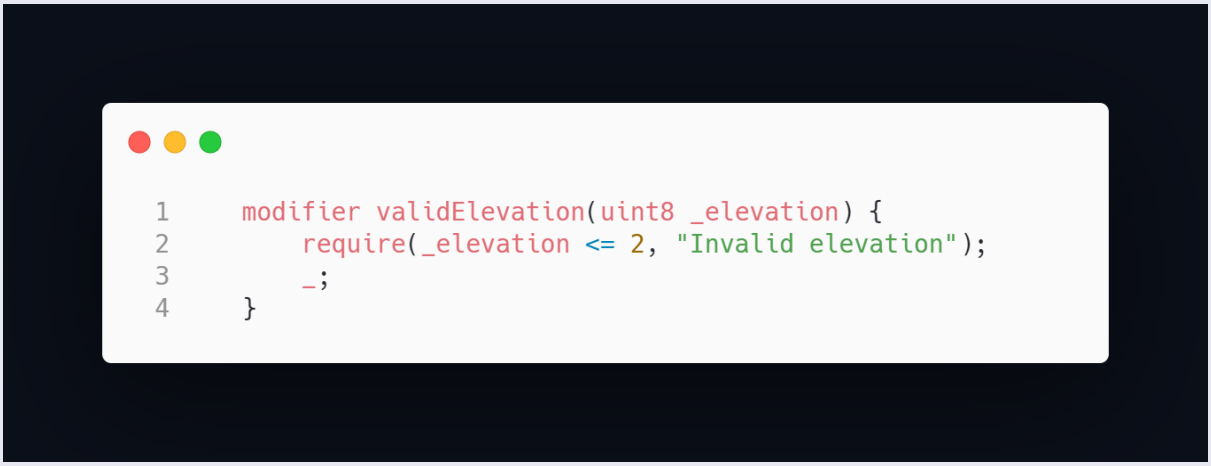
No Events Emitted For Changes To Protocol Values

FINDING ID	#0030
SEVERITY	Informational
STATUS	**Closed**
LOCATION	• ElevationHelper.sol -> 148-151: function setTrustedSeederAdd(address _trustedSeeder) external onlyCartographer • ElevationHelper.sol -> 405-408: function markWinningTotem(uint8 _elevation, uint8 _winner) internal • SummitReferrals.sol - > 60-65: function burnUnclaimedReferralRewardsAndRolloverRound(address referee) external onlyCartographer

DESCRIPTION	Functions that change important variables should include emit logs such that users can more easily monitor the change.
RECOMMENDATION	Add emit logs to these functions. Ensure that these values are secured via a timelock.
MITIGATED/COMMENT	Events were added. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Unused Modifiers

FINDING ID	#0031
SEVERITY	Informational
STATUS	**Closed**
LOCATION	CartographerElevation.sol -> 228-231



```
1    modifier validElevation(uint8 _elevation) {
2        require(_elevation <= 2, "Invalid elevation");
3        _;
4    }
```

DESCRIPTION	The *validElevation(uint8)* modifier is never used.
RECOMMENDATION	Remove the modifier.
MITIGATED/COMMENT	The modifier was removed. Reviewed in commit 7a89c5840bec5c74aa503709c796346ff75684ba

Potential Underflows

FINDING ID	#0032
SEVERITY	Medium Risk
STATUS	**Closed**
LOCATION	Non-exhaustive list of locations using unsafe math: Revised contracts 0ff22344d8caed8e1e04ebbd9807ef9c30a5651f CartographerElevation.sol -> 1075-1078 CartographerExpedition.sol -> 738-744 CartographerExpedition.sol -> 811-818

```
1         pool.totemSupplies[_prevTotem] -= user.staked;
2         pool.totemSupplies[_newTotem] += user.staked;
3         pool.totemRoundRewards[_prevTotem] -= user.roundRew;
4         pool.totemRoundRewards[_newTotem] += user.roundRew;
```

```
1         if (_isDeposit) {
2             user.summitStaked += _summitAmount;
3             user.lpStaked += _lpAmount;
4         } else {
5             user.summitStaked -= _summitAmount;
6             user.lpStaked -= _lpAmount;
7         }
```

```
1         if (user.summitStaked > 0) {
2             pool.totemSummitSupply[userTotem[_userAdd]] -=
  user.summitStaked;
3             pool.totemSummitSupply[_totem] +=
  user.summitStaked;
4         }
5         if (user.lpStaked > 0) {
6             pool.totemLpSupply[userTotem[_userAdd]] -=
  user.lpStaked;
7             pool.totemLpSupply[_totem] += user.lpStaked;
8         }
```

DESCRIPTION	Solidity versions before 0.8.0 do not automatically check for overflow or underflow.
RECOMMENDATION	Use SafeMath wherever possible.
MITIGATED/COMMENT	Arithmetic operations were replaced with SafeMath. Reviewed in commit [bb1a005902d0df7d75b0584a5682812d27d6a564]

On-Chain Analysis

Timelock Delay Is Short

FINDING ID	#0041
SEVERITY	Medium Risk
STATUS	**Partially Mitigated**
LOCATION	Timelock 0xF2c7D7773AC581160B042C2da1eff75AA7618b54

DESCRIPTION	The timelock delay is set to 6 hours. Obelisk recommends a timelock delay for all functionality of at least 72 hours. Transactions to set a specific delay of 72 hours for the following functions were queued but not executed: • setExpedAdd(address) • setTokenPassthroughStrategy(address,address) • retireTokenPassthroughStrategy(address) Note that the timelock contract applies its delay to itself. As a result, bypassing the timelock delay can be done in 12 hours: 6 hours to set the specific delay to the minimum delay, then the next 6 to queue a previously unavailable transaction.
RECOMMENDATION	Set the general timelock delay to 72 hours.
MITIGATED/COMMENT	The timelock delay was set to 24 hours. The specific delays of the noted functions were set to 72 hours.

Timelock Admin Not Correctly Initialized

FINDING ID	#0042
SEVERITY	Medium Risk
STATUS	**Closed**
LOCATION	Timelock 0xF2c7D7773AC581160B042C2da1eff75AA7618b54

DESCRIPTION	The timelock contract has a modification that allows the admin to be set once without the appropriate delay. Once it is set, the *admin_initialized* flag ensures proper delays are enforced to later admin changes. However, since *Timelock.setPendingAdmin()* has never been called, this flag has not yet been set.
RECOMMENDATION	Set the timelock admin again to ensure it is correctly initialized.
MITIGATED/COMMENT	The project team has implemented the recommended fix.

Cartographer Expedition Address Is An EOA

FINDING ID	#0043
SEVERITY	Medium Risk
STATUS	**Open**
LOCATION	Cartographer 0x46d303b6829aDc7AC3217D92f71B1DbbE77eBBA2

DESCRIPTION	The value of *cartographer.expedAdd* and *cartographer.cartographerExpedition* are expected to be the same. However, *cartographer.expedAdd* is set to an externally owned account. This will result in the yields being sent to the EOA instead of the expedition as intended. Refer to Finding #0006. EOA Address: 0x2370b8Ff222e12CC95b0c455B904211A58c28d21
RECOMMENDATION	Update the expedition address using `cartographer.setExpedAdd()`.
MITIGATED/COMMENT	Project Team Comment: “The expedition treasury is intentionally left as an EOA on project launch. The Expedition Treasury accumulates different tokens and LPs from each farm we offer, however, we want the Expeditions to run with specific stablecoins or tokens with deep liquidity and consistent valuation. After launch one of the team's top priorities is to replace this EOA with a contract that is able to trustless break lps and swap tokens to the desired Expedition tokens. We will consult with the Summit DeFi community to determine whether this new Expedition Accumulator should be placed behind our timelock.”

Vaults With Withdrawal Fees Deployed

FINDING ID	#0044
SEVERITY	Low Risk
STATUS	**Open**
LOCATION	Cartographer 0x46d303b6829aDc7AC3217D92f71B1DbbE77eBBA2

DESCRIPTION	The underlying contracts of passthrough contracts are expected to have no withdrawal fee. Retiring a passthrough strategy with an underlying fee will cause the recorded tokens to become out of sync with the available tokens. The following fees were noted: • TOMB WFTM Spooky-LP 0.1% • TSHARE WFTM Spooky-LP 0.1% • BOO WFTM Spooky-LP 0.1% • USDC WFTM Spooky-LP 0.1% • ETH WFTM Spooky-LP 0.1% • fUSDT WFTM Spooky-LP 0.1% • DAI WFTM Spooky-LP 0.1% • USDT/DAI/USDC Curve 0.01% • BIFI 0.05% Refer to Finding #0007.
RECOMMENDATION	Don't retire the existing passthrough strategies until they are fully drained.
MITIGATED/COMMENT	Project Team Comment: “Our Passthrough Strategy partner Beefy Finance has a maximum of 0.1% withdrawal fees across all their vaults. We will never pass through to a vault with a withdrawal fee higher than 0.2%. The farm fees offered by Summit DeFi will never be reduced below the 3 times the withdrawal fee of the underlying Passthrough Strategy vault to ensure that in the event we need to retire a strategy, we retain a withdrawal fee buffer, ensuring all users are able to withdraw their expected amount.”

Appendix A - Reviewed Documents

Document	Address
libs/BEP20.sol	N/A
libs/IBEP20.sol	N/A
libs/MockBEP20.sol	N/A
libs/Multicall.sol	N/A
libs/SafeBEP20.sol	N/A
BeefyVaultV2Passthrough.sol	N/A
BeefyVaultV6NativePassthrough.sol	N/A
BeefyVaultV6Passthrough.sol	TOMB WFTM Spooky-LP 0xc2D2A8E5237D17AbD9313dc17b5Ab2A9b8c8C019 TSHARE WFTM Spooky-LP 0x67EC9d9e521B2a3E6f10c323fF78a25D9C227D82 BOO WFTM Spooky-LP 0x80Ad389deEF1896c6ebB62B3ecc9AC7Bea70f6E3 USDC WFTM Spooky-LP 0x78Ce677f2484D7437f582BcAA444978554947504 ETH WFTM Spooky-LP 0x6bee330E62ec0eFD1F82c56f45fba462042bDcF5 fUSDT WFTM Spooky-LP 0x451a1B1d1C231beD8f99151486F12223D8C298dF DAI WFTM Spooky-LP 0x1E44b134946cF3aE1Fd914E2Ed1afEF354Af13F0 USDT/DAI/USDC Curve 0x6Aaa615FBb3f4a6Ac40A5280EbF2C9db0AbDFDef USDC 0x431a593671778a50cF5fC9406df50128F4137d04 fUSDT

	0x572021c29a1BAA1E722B4138F88d58564EFCB2ea DAI 0x60841AB003A50184757e4A987D29B0Cb0D2B92D9 BIFI 0x33F33A22E3a65c587d8C4B63727d6305B4983aC4
Cartographer.sol	0x46d303b6829aDc7AC3217D92f71B1DbbE77eBBA2
CartographerElevation.sol	0xdE1e14e2ED8B2D883B8338b514dDc173e792271a
CartographerExpedition.sol	0xb6067591A9b25c4A880F622e853fEd71016F5b05
CartographerOasis.sol	0x68889c9d8e923b3e310B60ee588242A407fa6755
ElevationHelper.sol	0xb3dC2531cbe58c4C23454D0c13bc572E16C60F3a
IPassthrough.sol	N/A
ISubCart.sol	N/A
MasterChefPassthrough.sol	N/A
SummitReferrals.sol	0x0B90dd88692Ec4fd4A77584713E3770057272B38
SummitToken.sol	0x8F9bCCB6Dd999148Da1808aC290F2274b13D7994
Timelock.sol	0xF2c7D7773AC581160B042C2da1eff75AA7618b54

Revisions

https://github.com/summit-defi/summit-contracts

- Revision 1: 877cbd7d7d72ea346ecf9f2a970e0d6caa47eff6
- Revision 2: 7a89c5840bec5c74aa503709c796346ff75684ba
- Revision 3: 0ff22344d8caed8e1e04ebbd9807ef9c30a5651f
- Revision 4: bb1a005902d0df7d75b0584a5682812d27d6a564
- Revision 5: 163c2dbe10b03d5f1d95d7d129235fbade69d140
- Revision 6: e352eff14bc5d0855d0f6fc8388b6722f0fd69a1
- Revision 7: a834cf2787dd25e4d41d6c44489d2a41eceaa4a7
- Revision 8: 86d4a3d3dc2e90d722ab0208d2a73d8b4dda998d
- Revision 9: cd072c2a56c85674cc04e7f54bc2937442029ee2
- Revision 10: 2111be7899f0d956f1af7391966af2440b6a0a8f
- Revision 11: 037c84ef7880433a1be661efbc82c6679c77f7c5

Imported Contracts

OpenZeppelin: 3.1.0

Externally Owned Accounts

0x3a7679E3662bC7c2EB2B1E71FA221dA430c6f64B - Developer
0x7E1e4354de68B644c30b40F983f66aF60042fF69 - Trusted Seeder
0x2370b8Ff222e12CC95b0c455B904211A58c28d21 - cartographer.expedAdd

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 - Testing Standard

An ordinary audit is conducted using these steps.

1. Gather all information
2. Conduct a first visual inspection of documents and contracts
3. Go through all functions of the contract manually (2 independent auditors)
    a. Discuss findings
4. Use specialized tools to find security flaws
    a. Discuss findings
5. Follow up with project lead of findings
6. If there are flaws, and they are corrected, restart from step 2
7. Write and publish a report

During our audit, a thorough investigation has been conducted employing both automated analysis and manual inspection techniques. Our auditing method lays a particular focus on the following important concepts:

- Ensuring that the code and codebase use best practices, industry standards, and available libraries.
- Testing the contract from different angles ensures that it works under a multitude of circumstances.
- Analyzing the contracts through databases of common security flaws.

OBELISK

Part of Tibereum Group