OBELISK

Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 61	2022-01-23	Zapmore, @DoD4uFN	Audit Draft

Audit Notes

Audit Date	2021-12-05 - 2022-01-22
Auditor/Auditors	@DoD4uFN, @mechwar
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	OB512158349

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

Name	Raven
Description	Multi-chain yield optimizer.
Website	https://raven.moe/
Contact	https://twitter.com/RavenDotMoe
Contact information	@Beastly418 on TG
Token Name(s)	N/A
Token Short	N/A
Contract(s)	See Appendix A
Code Language	Solidity
Chain	Polygon

Audit of Raven

> **Even though there were many initial issues found, the Raven team worked to close or mitigate all issues besides one.**

Obelisk was commissioned by Raven on the 3rd of December 2021 to conduct a comprehensive audit of Ravens' contracts. The following audit was conducted between the 5th of December 2021 and the 22nd of January 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.

During Obelisks' audit of Raven contracts, the auditors found multiple vulnerabilities, in all risk levels. The auditors gave suggestions on possible fixes to the issues found. The Raven Dev team worked to solve or mitigate these issues found. The only issue that is still fully open is Issue #9 which refers to the slippage limit, or lack of it, which makes it possible to front-run the swap that the contract makes, which could make the swap end up with an unfavorable exchange rate of the reward.

Low-Risk Issue #38 is partially closed as a lock was added to make it work as it should, however need to make sure to not call notifyRewardAmount() once the boost contract is disconnected.

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 Raven project**.**

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

Summary Table

Finding	ID	Severity	Status
Owner Can Change Router and Vault Address	#0001	High Risk	Closed
Approval Delay Of Strategy Upgrade Is Zero	#0002	High Risk	Mitigated
Boost Contract Setter Has No Delay	#0003	High Risk	Closed
Strategy PID Can Be Set To Any Value	#0004	Medium Risk	Closed
Proposed Strategy Can Overwrite Any Strategy	#0005	Medium Risk	Closed
No Limit For Protocol Values	#0006	Medium Risk	Closed
Protocol Values That Govern Reward Distribution Should Not Be Modifiable	#0007	Medium Risk	Closed
Prevent Withdraw Of Reward Token	#0008	Medium Risk	Closed
Swapping Tokens Can Be Frontrun	#0009	Medium Risk	Open
Potential Reentrancy	#0010	Medium Risk	Closed
All Tokens Should Be Transferred When Retiring An Older Strategy	#0011	Medium Risk	Closed
Claiming Rewards Does Not Correctly Check For Available Balance	#0012	Low Risk	Closed
Setting Of Set Boosted Flag Should Check For Valid Boost Contract	#0013	Low Risk	Closed
Strategy Functions To Deposit And Claim Rewards Should Be Restricted	#0014	Low Risk	Closed
Off By One Indexing Of User Deposit Harvest Array	#0015	Low Risk	Closed

Uniswap Routes Not Updated Upon Strategy Construction	#0016	Low Risk	Closed
Uninitialized Protocol Value	#0017	Low Risk	Closed
Strategy Does Not Swap To Want Token	#0018	Low Risk	Closed
SendRewardsToVault Doesn't Transfer All The Balance	#0019	Informational	Closed
Use Of tx.origin	#0020	Informational	Closed
Unbound Loop	#0021	Informational	Closed
Outdated Version Of Reentrancy Guard	#0022	Informational	Open
Missing Zero Checks	#0023	Informational	Closed
No Events Emitted For Changes To Protocol Values	#0024	Informational	Closed
Duplicate Contract Naming	#0025	Informational	Closed
Unmatched File And Contract Name	#0026	Informational	Closed
Incorrect Or Missing Imports	#0027	Informational	Closed
Unused Variables	#0028	Informational	Closed
Boost Contract Should Be Set By The Proposed Boost Contract	#0029	Low Risk	Closed
Proposed Strategy Should Cross Check Strategy Want vs. Vault's Reward Token	#0030	Low Risk	Closed
Different Strategies Could Have Same Want Token	#0031	Medium Risk	Closed
Proposed Strategy Could Have Different Want Token	#0032	Medium Risk	Closed
Strategy Doesn't Deposit Want Token When Harvesting	#0033	Low Risk	Closed
Protocol Will Malfunction When The Amount Staked LP Is Zero	#0034	Medium Risk	Closed

Vault And Strategy Can Have Different Boost Contracts	#0035	High Risk	Closed
Rounding Errors Can Prevent Users From Claiming Rewards	#0036	Medium Risk	Closed
Changing Boost Contract Prevents Claiming Rewards	#0037	Low Risk	Closed
Changing Boost Contract Can Desynchronize Deposit Amounts From The Vault	#0038	Low Risk	Partially Closed
Unverified Contract	#0039	High Risk	Closed
Privileged Addresses Are EOA	#0040	Informational	Closed

Findings

Manual Analysis

Owner Can Change Router and Vault Address

FINDING ID	#0001
SEVERITY	High Risk
STATUS	Closed
LOCATION	StratManager.sol -> 77-79

```
1    function setUnirouter(address _unirouter) external onlyOwner {
2        unirouter = _unirouter;
3    }
```

LOCATION	StratManager.sol -> 85-87

```
1    function setVault(address _vault) external onlyOwner {
2        vault = _vault;
3    }
```

DESCRIPTION	The vault strategies use a uniswap router to exchange tokens. A malicious actor as the owner can change the router to a malicious contract. A malicious actor as the owner can also change the vault to a malicious contract. As the router and vault have the approval to withdraw reward tokens from the strategy, this can be used to drain the entire reward and staking balance.
RECOMMENDATION	Set the router and vault path a single time during initialization.
RESOLUTION	The project team has implemented the recommended fix by removing the setter functions. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Approval Delay Of Strategy Upgrade Is Zero

FINDING ID	#0002
SEVERITY	High Risk
STATUS	Mitigated
LOCATION	• VampireVaultMultiV3 -> 49: *uint256 public constant approvalDelay = 0;*

DESCRIPTION	The value of *approvalDelay* is zero, which means that a bad actor could upgrade any strategy instantly. The new strategy could contain malicious code and result in a drain of user funds.
RECOMMENDATION	Increase the delay to at least 72 hours.
RESOLUTION	The approval delay is set at the constructor, a lower bound is recommended to be used. The team added a 72-hour approval delay at deployment. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Boost Contract Setter Has No Delay

FINDING ID	#0003
SEVERITY	High Risk
STATUS	Closed
LOCATION	VampireVaultMultiV3.sol -> 341-346

```
1     function setBoostContract(address _boostCon, bool _boost, uint256
  pid) public onlyOwner {
2         setStratBoosted(_boost, pid);
3         boostContract[pid] = _boostCon;
4         //strategies[pid].setBoostContract(_boostCon);
5         //Add in the boostContract to strategy here
6     }
```

DESCRIPTION	The function *setBoostContract()* has no delay at setting a Boost Contract. A malicious actor can set a boost contract containing vulnerable code, when a user deposits, instead of depositing to Strategy contract, the funds will first be sent to Boost Contract, where they can be drained.
RECOMMENDATION	Make sure to deploy a timelock of at least 72h.
RESOLUTION	The project team has implemented the recommended fix by adding a *proposeBoost()* function. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Strategy PID Can Be Set To Any Value

FINDING ID	#0004
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	AbstractVampireStrategyV1 -> 32-48

```
1     constructor(
2         address _lpToken,
3         address _want,
4         address _stakingContract,
5         address _vault,     //Vault that owns this strategy
6         address _unirouter, //Main router we use for converting
  reward to native (ex: SushiRouter if we go from sushi to matic)
7         address _keeper,
8         address _strategist,
9         address _ravenTreasury,
10        uint256 _pid
11    ) StratManager(_keeper, _strategist, _unirouter, _vault,
  _ravenTreasury) public {
12        lpToken = _lpToken; // lp token we need to deposit to the
  stakingContract
13        want = _want;       // token we 'want' to get in return (AC
  receipt Token)
14        stakingContract = _stakingContract; // where we deposit the
  LP tokens
15        _giveAllowances();  // gives the necessary routers
  permissions to spend the tokens
16        pid = _pid;         // this strategy's pool id in it's vault
  [DO NOT MESS THIS UP]
17    }
```

DESCRIPTION	The *pid* of a strategy can be set to any value. If that happens, it can lead to loss of protocol functionality.
RECOMMENDATION	Introduce a factory for the Strategies in order to avoid setting the *pid* manually.
RESOLUTION	The project team has implemented the recommended fix by adding the appropriate pid checks when interacting with the *VampireVaultMultiV3* vault contract. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Proposed Strategy Can Overwrite Any Strategy

FINDING ID	#0005
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	VampireVaultMultiV3.sol -> 380-392

```
1     function upgradeStrat(uint256 pid) public onlyOwner {
2         require(stratCandidate.implementation != address(0), "There
  is no candidate");
3         require(stratCandidate.proposedTime.add(approvalDelay) <
  block.timestamp, "Delay has not passed");
4
5         emit UpgradeStrat(stratCandidate.implementation);
6
7         strategies[pid].retireStrat();
8         strategies[pid] = IStrategy(stratCandidate.implementation);
9         stratCandidate.implementation = address(0);
10        stratCandidate.proposedTime = 5000000000;
11
12        earn(pid);
13    }
```

DESCRIPTION	The function *upgradeStrat()* can be called with any *pid* as a parameter. Which can lead to upgrading the wrong strategy.
RECOMMENDATION	Add a require statement that check if the *pid* is equal to *stratCandidate*'s *pid*.
RESOLUTION	The project team has implemented the recommended fix by adding a *require* statement in the *proposeStrat()* function that makes sure that the *pid* used in the *proposeStrat()* is aligned with the strategy contract itself. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

No Limit For Protocol Values

FINDING ID	#0006
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	SingleStakingBoost.sol -> 36-46

```
1    constructor(
2        address _rewardsToken,
3        address _stakingToken,
4        uint256 _rewardsDuration,
5        address _vampStrategy
6    ) {
7        rewardsToken = IERC20(_rewardsToken);
8        stakingToken = IERC20(_stakingToken);
9        rewardsDuration = _rewardsDuration;
10       strategy = IVampStrategy(_vampStrategy);
11   }
```

LOCATION	SingleStakingBoost.sol -> 131-150

```
1     function notifyRewardAmount(uint256 reward) external onlyOwner
  updateReward(address(0)) {
2         if (block.timestamp >= periodFinish) {
3             rewardRate = reward.div(rewardsDuration);
4         } else {
5             uint256 remaining = periodFinish.sub(block.timestamp);
6             uint256 leftover = remaining.mul(rewardRate);
7             rewardRate = reward.add(leftover).div(rewardsDuration);
8         }
9
10        // Ensure the provided reward amount is not more than the
  balance in the contract.
11        // This keeps the reward rate in the right range, preventing
  overflows due to
12        // very high values of rewardRate in the earned and
  rewardsPerToken functions;
13        // Reward + leftover must be less than 2^256 / 10^18 to avoid
  overflow.
14        uint balance = rewardsToken.balanceOf(address(this));
15        require(rewardRate <= balance.div(rewardsDuration), "Provided
  reward too high");
16
17        lastUpdateTime = block.timestamp;
18        periodFinish = block.timestamp.add(rewardsDuration);
19        emit RewardAdded(reward);
20    }
```

DESCRIPTION	The value of *rewardsDuration* can be arbitrarily high, causing the *rewardRate* being effectively zero. With the *notifyRewardAmount* function, reward could be set to be less than *rewardsDuration* which would mean that the *rewardRate* would also be zero.
RECOMMENDATION	Add an upper bound to the *rewardsDuration* in the constructor. Add a bound such that *reward* must be greater than *rewardsDuration* in the *notifyRewardAmount* function.
RESOLUTION	The project team has implemented the recommended fix. The bound in *notifyRewardAmount()* should be effective enough to prevent *rewardRate* from being zero. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Protocol Values That Govern Reward Distribution Should Not Be Modifiable

FINDING ID	#0007
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	SingleStakingBoost.sol -> 153-155

```
1     function updatePeriodFinish(uint timestamp) external onlyOwner
  updateReward(address(0)) {
2         periodFinish = timestamp;
3     }
```

LOCATION	SingleStakingBoost.sol -> 173-175

```
1     function updateLastTime(uint timestamp) external onlyOwner {
2         lastUpdateTime = timestamp;
3     }
```

DESCRIPTION	The values of *periodFinish* and *lastUpdateTime* are set to govern the reward distribution. These values should not be modifiable to prevent incorrect reward distribution. A malicious actor as the owner could set *lastUpdateTime* to a low number in order to drain all reward tokens from the contract.
RECOMMENDATION	Remove these only owner functions.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Prevent Withdraw Of Reward Token

FINDING ID	#0008
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	SingleStakingBoost.sol -> 158-162

```
1     function recoverERC20(address tokenAddress, uint256 tokenAmount)
  external onlyOwner {
2         require(tokenAddress != address(stakingToken), "Cannot
  withdraw the staking token");
3         IERC20(tokenAddress).safeTransfer(owner(), tokenAmount);
4         emit Recovered(tokenAddress, tokenAmount);
5     }
```

DESCRIPTION	The reward token could be withdrawn with the *recoverERC20* function.
RECOMMENDATION	Add a check to prohibit the withdrawal of the reward token.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Swapping Tokens Can Be Frontrun

FINDING ID	#0009
SEVERITY	Medium Risk
STATUS	Open

LOCATION	• BaseVampStrategy.sol -> 63: *IUniswapRouterETH(unirouter).swapExactTokensForTokens(rewardBal, 0, rewardToMatic, address(this), now); //Convert Quick into matic*
	• BaseVampStrategy.sol -> 67: *IUniswapRouterETH(elkRouter).swapExactTokensForTokens(maticBal, 0, wantPath1, address(this), now); //Convert matic into elk*
	• BaseVampStrategy.sol -> 77: *IUniswapRouterETH(unirouter).swapExactTokensForTokens(toMatic, 0, rewardToMatic, address(this), now);*

DESCRIPTION	Tokens are exchanged via a DEX router but do not provide a slippage limit. The rewards from the swap can be front-run.
RECOMMENDATION	Provide a slippage limit for the token as a parameter or use an oracle's time-weighted average price (TWAP) in order to prevent frontrunning.
RESOLUTION	The project team has not implemented the recommended fix. Reviewed in commit 8c2053002cedfb9b9176f006658dd57e643657dd

Potential Reentrancy

FINDING ID	#0010
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	• VampireVaultMultiV3.sol -> 210: `function withdraw(uint256 pid, uint256 _amount) public {` • VampireVaultMultiV3.sol -> 226: `function emergencyWithdraw(uint256 pid, uint256 _amount) public {` • VampireVaultMultiV3.sol -> 91: `function claim(uint256 pid) public`

DESCRIPTION	The *withdraw()* function of the vault needs to be non-reentrant. If the underlying token is vulnerable to reentrancy then this function could be re-entered and drain a user's balance. The *claim()* function of the vault needs to be non-reentrant. If the underlying token is vulnerable to reentrancy then this function could be re-entered and drain the vault rewards.
RECOMMENDATION	Add the *nonReentrant* modifier to the *withdraw()*, *emergencyWithdraw()* and *claim()* functions. Alternatively, apply the checks-effects-interactions pattern to them.
RESOLUTION	The project team has implemented the recommended fix by adding the *nonReentrant* modifier. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

All Tokens Should Be Transferred When Retiring An Older Strategy

FINDING ID	#0011
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	VampireVaultMultiV3.sol -> 380-392

```
1     function upgradeStrat(uint256 pid) public onlyOwner {
2         require(stratCandidate.implementation != address(0), "There
  is no candidate");
3         require(stratCandidate.proposedTime.add(approvalDelay) <
  block.timestamp, "Delay has not passed");
4
5         emit UpgradeStrat(stratCandidate.implementation);
6
7         strategies[pid].retireStrat();
8         strategies[pid] = IStrategy(stratCandidate.implementation);
9         stratCandidate.implementation = address(0);
10        stratCandidate.proposedTime = 5000000000;
11
12        earn(pid);
13    }
```

DESCRIPTION	All tokens should be transferred to the new strategy when the older strategy is retired. This includes the reward and want tokens. This will prevent funds from being locked in the prior strategy.
RECOMMENDATION	Add functionality to transfer all applicable tokens to the new strategy to avoid loss of funds.
RESOLUTION	The project team has implemented the recommended fix by adding the *harvest()* function call before retiring the strategy. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Claiming Rewards Does Not Correctly Check For Available Balance

FINDING ID	#0012
SEVERITY	Low Risk
STATUS	Closed
LOCATION	SingleStakingBoost.sol -> 144-145

```
1         uint balance = rewardsToken.balanceOf(address(this));
2         require(rewardRate <= balance.div(rewardsDuration), "Provided
  reward too high");
```

DESCRIPTION	The function *notifyRewardAmount()* does not account for tokens already assigned for distribution when checking that enough tokens are available.
RECOMMENDATION	Ensure that tokens already assigned for distribution to staked users are not included in the balance check. NOTE: When comparing *rewardRate* to the current balance, the rewards that have been assigned to users are not taken into account. Introduce a *rewardsAssigned* variable that is increased at *updateReward()* (when rewards are assigned), and decreased at *getReward()* (when rewards are withdrawn). Then subtract *rewardsAssigned* from *balance* in line 145. Also, when *rewardsToken* == *stakingToken*, the *_totalSupply* should be subtracted from the balance in line 144.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 8c2053002cedfb9b9176f006658dd57e643657dd

Setting Of Set Boosted Flag Should Check For Valid Boost Contract

FINDING ID	#0013
SEVERITY	Low Risk
STATUS	Closed
LOCATION	VampireVaultMultiV3.sol -> 337-346

```
    function setStratBoosted(bool _boost, uint256 pid) public onlyOwner {
        strategies[pid].setBoosted(_boost);
    }

    function setBoostContract(address _boostCon, bool _boost, uint256 pid) public onlyOwner {
        setStratBoosted(_boost, pid);
        boostContract[pid] = _boostCon;
        //strategies[pid].setBoostContract(_boostCon);
        //Add in the boostContract to strategy here
    }
```

DESCRIPTION	The *setStratBoosted()* function should not be callable if there is no valid boost contract.
RECOMMENDATION	In the *setStratBoosted()* function, add a check to make sure that *boostContract[pid]* is a valid contract address.
RESOLUTION	The project team has implemented the recommended fix by adding a check when proposing the boost contract. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Strategy Functions To Deposit And Claim Rewards Should Be Restricted

FINDING ID	#0014
SEVERITY	Low Risk
STATUS	Closed
LOCATION	AbstractVampireStrategyV1.sol -> 57-67

```
1    // puts the funds to work
2    function deposit() public whenNotPaused {
3        uint256 wantBal = IERC20(lpToken).balanceOf(address(this));
4        if (wantBal > 0) {
5            IRewardPool(stakingContract).stake(wantBal);
6        }
7    }
8
9    function claimRewards() public {
10       IRewardPool(stakingContract).getReward();
11   }
```

DESCRIPTION	The deposit and claiming of rewards should be controlled by the vault/boost contract likewise with the *withdraw()* function.
RECOMMENDATION	Add a restriction similar to the *withdraw()* function for the deposit and claiming of rewards function.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Off By One Indexing Of User Deposit Harvest Array

FINDING ID	#0015
SEVERITY	Low Risk
STATUS	Closed
LOCATION	VampireVaultMultiV3.sol -> 110

```
1 for (uint256 i = getUserDepositHarvest(pid, _depositor); i <
  harvestPerLP[pid].length; i++) {
```

LOCATION	VampireVaultMultiV3.sol -> 251-253

```
1     function getUserDepositHarvest(uint256 pid, address addy) public
  view returns (uint256) {
2         return users[addy][pid].depositHarvest;
3     }
```

DESCRIPTION	The *depositHarvest* represents the length of the array. However in the for loop, it should start at one minus the length of the array. Otherwise the loop is starting from the next harvest amounts record.
RECOMMENDATION	Subtract one from *getUserDepositHarvest()* since that should represent the correct starting index of the harvest amounts.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 8c2053002cedfb9b9176f006658dd57e643657dd

Uniswap Routes Not Updated Upon Strategy Construction

FINDING ID	#0016
SEVERITY	Low Risk
STATUS	Closed
LOCATION	BaseVampStrategy.sol -> 31-32

```
1     address[] public rewardToMatic = [reward, matic];
2     address[] public rewardToEth = [reward, eth];
```

DESCRIPTION	When the reward address is updated in the constructor these routes won't be updated.
RECOMMENDATION	Update the routes in the constructor.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Uninitialized Protocol Value

FINDING ID	#0017
SEVERITY	Low Risk
STATUS	Closed
LOCATION	• AbstractVampireStrategyV1.sol -> 21: *address public ACVault;*

DESCRIPTION	*ACVault* is never initialized but used in contracts.
RECOMMENDATION	Set the *ACVault* address in the *AbstractVampireStrategyV1* constructor.
RESOLUTION	The project team has implemented the recommended fix. *ACVault* address was moved to the *BaseVampStrategy* contract. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Strategy Does Not Swap To Want Token

FINDING ID	#0018
SEVERITY	Low Risk
STATUS	Closed
LOCATION	BaseVampStrategy.sol -> 58-71

```
1     //Converts the reward tokens into the output token we're looking
  for
2     function swapRewardsToWant() override internal {
3         //Change this as needed for different strategiess
4         // Swap to matic
5         uint256 rewardBal = IERC20(reward).balanceOf(address(this));
6
   IUniswapRouterETH(unirouter).swapExactTokensForTokens(rewardBal, 0,
  rewardToMatic, address(this), now); //Convert Quick into matic
7
8         //Swap to elk
9         uint256 maticBal = IERC20(matic).balanceOf(address(this));
10
   IUniswapRouterETH(elkRouter).swapExactTokensForTokens(maticBal, 0,
  wantPath1, address(this), now); //Convert matic into elk
11
12        //Deposit into ElkSSVault
13
   IACVault(ACVault).deposit(IERC20(elk).balanceOf(address(this)));
14    }
```

DESCRIPTION	The reward balance is swapped to elk, but the elk is not swapped to the want token.
RECOMMENDATION	Add another swap using the *wantPath2* route after swapping to elk.
RESOLUTION	The project team has implemented the correct solution by swapping to the *want* token at the end of *swapRewardsToWant()*. This was done by updating the *wantPath1* path. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

SendRewardsToVault Doesn't Transfer All The Balance

FINDING ID	#0019
SEVERITY	Informational
STATUS	Closed
LOCATION	AbstractVampireStrategyV1.sol -> 99-102

```
1     function sendRewardsToVault() internal {
2         IVault(vault).depositRewards(pid, balanceOfWant().sub(10));
  // write how much we're sending to the vault
3         IERC20(want).safeTransfer(vault, balanceOfWant().sub(10));
   // send the reward tokens to the vault
4     }
```

DESCRIPTION	*sendRewardsToVault()* always subtracts the balance of want by 10. That way there is always some dust left in the contract. Based on the decimals of the want tokens, these 10 tokens could be a significant amount. If this is intended, there is a risk of a malicious actor withdrawing these leftover tokens using *inCaseTokensGetStuck()*.
RECOMMENDATION	Remove this subtraction.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Use Of tx.origin

FINDING ID	#0020
SEVERITY	Informational
STATUS	Closed
LOCATION	• StratManager.sol -> 52: *require(msg.sender == tx.origin, "!EOA");* • AbstractVampireStrategyV1.sol -> 90: *if (tx.origin == owner() || paused()) {*

DESCRIPTION	Be aware that tx.origin might not stay useful in future updates, Vitalik: "Do NOT assume that tx.origin will continue to be usable or meaningful."
RECOMMENDATION	Be aware of the downsides of using tx.origin and watch out for changes concerning tx.origin.
RESOLUTION	The project team has implemented the recommended fix by removing the setter functions. Reviewed in commit 937e4131e9e50e6e11a2ecf395f551223496db15

Unbound Loop

FINDING ID	#0021
SEVERITY	Informational
STATUS	Closed
LOCATION	VampireVaultMultiV3.sol -> 419-421

```
        for(i = 0; i < strategies.length; i++) {
            require(_token != address(LPToken(i)), "!token");
        }
```

DESCRIPTION	Unbound loops may revert due to the gas fee limit.
RECOMMENDATION	Add an upper bound to the strategies array length.
RESOLUTION	The project team has implemented a fix in the *claim()* function. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Outdated Version Of Reentrancy Guard

FINDING ID	#0022
SEVERITY	Informational
STATUS	Open
LOCATION	ReentrancyGuard.sol

DESCRIPTION	*ReentrancyGuard.sol* is an outdated version of the OpenZeppelin implementation.
RECOMMENDATION	Directly import the latest implementation.
RESOLUTION	The project team has decided to stick with the older version of Reentrancy Guard. It is still recommended to upgrade the Reentrancy Guard contract. Reviewed in commit 8c2053002cedfb9b9176f006658dd57e643657dd

Boost Contract Should Be Set By The Proposed Boost Contract

FINDING ID	#0029
SEVERITY	Low Risk
STATUS	Closed
LOCATION	VampireVaultMultiV3.sol -> 391-400

```
1     function boostStrategy(address _boostCon, bool _boost, uint256
  pid) public onlyOwner {
2         require(boostCandidate[pid].implementation != address(0),
  "There is no candidate");
3         require(boostCandidate[pid].proposedTime.add(approvalDelay) <
  block.timestamp, "Delay has not passed");
4         emit BoostStrat(boostCandidate[pid].implementation);
5         boostCandidate[pid].implementation = address(0);
6         boostCandidate[pid].proposedTime = 5000000000;
7
8         setStratBoosted(_boost, pid);
9         boostContract[pid] = _boostCon;
10    }
```

DESCRIPTION	The *boostStrategy()* function call uses the input parameter *_boostCon* instead of the proposed boost contract address stored in the *boostCandidate[pid].implementation* value.
RECOMMENDATION	Remove the input parameter *_boostCon* from the *boostStrategy()* function and the *boostContract[pid]* to be the proposed boost candidate. Move *setStratBoosted()* and the assignment of boostContract before resetting the *boostCandidate[pid].implementation*.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 8c2053002cedfb9b9176f006658dd57e643657dd

Proposed Strategy Should Cross Check Strategy Want vs. Vault's Reward Token

FINDING ID	#0030
SEVERITY	Low Risk
STATUS	Closed
LOCATION	VampireVaultMultiV3.sol -> 409-423

```
1     function addStrategy(address strategy) public onlyOwner {
2         require(strategies.length == IStrategy(strategy).pid(), "PID
  doesn't match");
3         require(address(IStrategy(strategy).want()) == reward,
  "reward token doesn't match the strategy want");
4         strategies.push(IStrategy(strategy));
5
6         //Initialize the arrays to have a 0x00 candidate
7         boostCandidate.push(ContractCandidate({
8             implementation: address(0),
9             proposedTime: 5000000000
10        }));
11        stratCandidate.push(ContractCandidate({
12            implementation: address(0),
13            proposedTime: 5000000000
14        }));
15    }
```

DESCRIPTION	The *addStrategy()* function checks that the proposed strategy's want token and the vault's reward token are the same. Although, this check should be at proposeStrat(). NOTE: Requiring strategies want the token to be equal to vault's reward token seems like a typo.
RECOMMENDATION	Add a similar check that is present in the *addStrategy()* function.
RESOLUTION	Team comment: *Want in our strategies is the final token the strategy gets that it sends back to the vault for claiming. Working as intended.* With this realization, the project team has implemented the recommended fix.

	Reviewed in commit 8c2053002cedfb9b9176f006658dd57e643657dd

Different Strategies Could Have Same Want Token

FINDING ID	#0031
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	VampireVaultMultiV3.sol -> 409-423

```
1     function addStrategy(address strategy) public onlyOwner {
2         require(strategies.length == IStrategy(strategy).pid(), "PID
  doesn't match");
3         require(address(IStrategy(strategy).want()) == reward,
  "reward token doesn't match the strategy want");
4         strategies.push(IStrategy(strategy));
5
6         //Initialize the arrays to have a 0x00 candidate
7         boostCandidate.push(ContractCandidate({
8             implementation: address(0),
9             proposedTime: 5000000000
10        }));
11        stratCandidate.push(ContractCandidate({
12            implementation: address(0),
13            proposedTime: 5000000000
14        }));
15    }
```

DESCRIPTION	When a new strategy is added, it can have the same want token as a previous strategy.
RECOMMENDATION	Check the previous strategies for the same want token.
RESOLUTION	Team Comment: *All strategies need to have the same want token. Working as intended.* A check was added to ensure that the strategy want token is consistent with the vault. Reviewed in commit 8c2053002cedfb9b9176f006658dd57e643657dd

Proposed Strategy Could Have Different Want Token

FINDING ID	#0032
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	VampireVaultMultiV3.sol -> 431-440

```
1     function proposeStrat(address _implementation, uint256 pid)
  public onlyOwner {
2         require(address(this) == IStrategy(_implementation).vault(),
  "Proposal not valid for this Vault");
3         require(pid == IStrategy(_implementation).pid(), "Proposal
  PID doesn't match what we want to swap it with");
4         stratCandidate[pid] = ContractCandidate({
5             implementation: _implementation,
6             proposedTime: block.timestamp
7          });
8
9         emit NewStratCandidate(_implementation);
10    }
```

DESCRIPTION	When proposing a strategy, the want token does not necessarily have to be the same as the previous strategy.
RECOMMENDATION	Check that the proposed strategy has the same want token as the previous strategy.
RESOLUTION	Team Comment: *All strategies need to have the same want token. Working as intended.* A check was added to ensure that the strategy want token is consistent with the vault. Reviewed in commit 8c2053002cedfb9b9176f006658dd57e643657dd

Strategy Doesn't Deposit Want Token When Harvesting

FINDING ID	#0033
SEVERITY	Low Risk
STATUS	Closed
LOCATION	BaseVampStrategy.sol -> 60-72

```
1     function swapRewardsToWant() override internal {
2         //Change this as needed for different strategiess
3         // Swap to matic
4         uint256 rewardBal = IERC20(reward).balanceOf(address(this));
5
    IUniswapRouterETH(unirouter).swapExactTokensForTokens(rewardBal, 0,
  rewardToMatic, address(this), now); //Convert Quick into matic
6
7         //Swap to elk
8         uint256 maticBal = IERC20(matic).balanceOf(address(this));
9
    IUniswapRouterETH(elkRouter).swapExactTokensForTokens(maticBal, 0,
  wantPath1, address(this), now); //Convert matic into elk
10
11        //Deposit into ElkSSVault
12
    IACVault(ACVault).deposit(IERC20(elk).balanceOf(address(this)));
13    }
```

DESCRIPTION	The strategy deposits *elk* to the underlying farm instead of the *want* token.
RECOMMENDATION	Replace *elk* with *want* on the last line of *swapRewardsToWant()* function.
RESOLUTION	The *want* token is a receipt of a vault, which can be redeemed for *elk* tokens. The strategy sends these receipt tokens back to its vault to be redeemable from the user. Reviewed in commit 22eb40151fb765ca7894422fdca60566a9708cd3

Protocol Will Malfunction When The Amount Staked LP Is Zero

FINDING ID	#0034
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	VampireVaultMultiV3.sol -> 431-440

```
1         uint256 rewardsPerLP = amt.div(balanceOfStakedLP(pid));
```

LOCATION	VampireVaultMultiV3.sol -> 431-440

```
1         if(balanceOfStakedLP(pid) == 0) {
2             //Only happens when no one is in the pool to prevent harvesting nothing and so we never go negative on indexing a claim
3             totRewPerLP.push(0);
4         } else {
5             harvest(pid);       //Harvest the pool before you deposit so you're not joining in the middle of a harvest
6         }
```

LOCATION	VampireVaultMultiV3.sol -> 431-440

```
1         uint256 diffRewPerLP = getRewardsPerLP(pid, current).sub(getRewardsPerLP(pid, start));
```

DESCRIPTION	1) When the balance of staked LPs reaches zero, the *depositRewards()* function will revert due to the divide by zero. 2) Pushing zero to *totalRewardsPerLP* wipes out the previous rewards per LP. This will cause the *claimable()* function to revert since the *diffRewPerLP* will become negative.

	When these functions malfunction, it can cause reward tokens to become stuck.
RECOMMENDATION	Check the balance of staked LPs, if it's zero, assign *rewardsPerLP* to zero, or else the transaction will revert. In conclusion, *VampireVaultMultiV3.depositRewards()* shouldn't revert because the strategy won't be able to harvest.
RESOLUTION	The necessary checks were put in place to make sure that *totalRewardsPerLP* remains consistent. Reviewed in commit 22eb40151fb765ca7894422fdca60566a9708cd3

Vault And Strategy Can Have Different Boost Contracts

FINDING ID	#0035
SEVERITY	High Risk
STATUS	Closed
LOCATION	Rev 6 StratManager.sol -> 104-107

```
1     function setBoostContract(address _boostContract) external
  onlyOwner {
2         boostContract = _boostContract;
3         emit BoostContractChanged(_boostContract);
4     }
```

LOCATION	Rev 6 VampireVaultMultiV3.sol -> 403-411

```
1     function boostStrategy(bool _boost, uint256 pid) public onlyOwner
  {
2         require(boostCandidate[pid].implementation != address(0),
  "There is no candidate");
3         require(boostCandidate[pid].proposedTime.add(approvalDelay) <
  block.timestamp, "Delay has not passed");
4         setStratBoosted(_boost, pid);
5         boostContract[pid] = boostCandidate[pid].implementation;
6         emit BoostStrat(boostCandidate[pid].implementation);
7         boostCandidate[pid].implementation = address(0);
8         boostCandidate[pid].proposedTime = 5000000000;
9     }
```

DESCRIPTION	The *boostContract* in the StratManager is set manually within the strategy. This can be different from the boost contract set in the Vault for that strategy. This can cause malfunctions or prevent users from withdrawing.
RECOMMENDATION	Update the strategy's boosted contract at the same time that it is set in the vault.
RESOLUTION	The project team has implemented the recommended fix.

	Reviewed in commit 904efd045788b81838fe3ae4caf73540858d8309

Rounding Errors Can Prevent Users From Claiming Rewards

FINDING ID	#0036
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	Rev 6 VampireVaultMultiV3.sol -> 480-486

```
1     function percent(uint numerator, uint denominator, uint
  precision) public pure returns(uint quotient) {
2           // caution, check safe-to-multiply here
3         uint _numerator  = numerator * 10 ** (precision+1);
4         // with rounding of last digit
5         uint _quotient =  ((_numerator / denominator) + 5) / 10;
6         return ( _quotient);
7     }
```

DESCRIPTION	Rounding up can lead to more rewards being distributed than are available. Within the *percent()* function, the SafeMath library isn't being used. This can cause overflows.
RECOMMENDATION	Always round down in reward calculations. Use SafeMath for division and multiplications.
RESOLUTION	The *percent()* function was modified to always round down. Reviewed in commit 904efd045788b81838fe3ae4caf73540858d8309

Changing Boost Contract Prevents Claiming Rewards

FINDING ID	#0037
SEVERITY	Low Risk
STATUS	Closed
LOCATION	Rev 6 VampireVaultMultiV3.sol -> 138-145

```
    function claimBoost(uint256 pid) internal {
        //If the strat is no longer boosted but you have claimable rewards you should be able to claim
        if(getBoostContract(pid) != address(0)) {
            if(isStratBoosted(pid) || getBoostClaimable(pid, msg.sender) > 0) {
                IBoost(boostContract[pid]).getReward(msg.sender);
            }
        }
    }
```

LOCATION	Rev 6 SingleStakingBoost.sol -> 117-125

```
    function getReward( address _depositor) public nonReentrant onlyVault updateReward(_depositor) {
        uint256 reward = rewards[_depositor];
        if (reward > 0) {
            rewards[_depositor] = 0;
            rewardsAssigned = rewardsAssigned.sub(reward);
            rewardsToken.safeTransfer(_depositor, reward);
            emit RewardPaid(_depositor, reward);
        }
    }
```

DESCRIPTION	Boost rewards can only be claimed from the boost contract via the vault. Therefore, removing the boosted contract will prevent users from claiming boosted rewards.
RECOMMENDATION	Ensure that the boost contract is not disconnected until the boosted rewards are completely distributed or allow users to claim rewards directly from the boost contract.

RESOLUTION	The contract was modified so that users can get rewards from the boost contract directly if it's disconnected. Reviewed in commit 904efd045788b81838fe3ae4caf73540858d8309

Changing Boost Contract Can Desynchronize Deposit Amounts From The Vault

FINDING ID	#0038
SEVERITY	Low Risk
STATUS	Partially Closed
LOCATION	Rev 6 SingleStakingBoost.sol -> 77-79

```
1     function earned(address account) public view returns (uint256) {
2         return
  _balances[account].mul(rewardPerToken().sub(userRewardPerTokenPaid[ac
  count])).div(1e18).add(rewards[account]);
3     }
```

LOCATION	Rev 6 VampireVaultMultiV3.sol -> 403-411

```
1     function boostStrategy(bool _boost, uint256 pid) public onlyOwner
  {
2         require(boostCandidate[pid].implementation != address(0),
  "There is no candidate");
3         require(boostCandidate[pid].proposedTime.add(approvalDelay) <
  block.timestamp, "Delay has not passed");
4         setStratBoosted(_boost, pid);
5         boostContract[pid] = boostCandidate[pid].implementation;
6         emit BoostStrat(boostCandidate[pid].implementation);
7         boostCandidate[pid].implementation = address(0);
8         boostCandidate[pid].proposedTime = 5000000000;
9     }
```

DESCRIPTION	If a boost contract is disconnected and later reconnected, the staked amounts recorded in the boost contract records can be different from those in the vault contract. During deposits and withdraws, the vault updates its internal *user.depositAmount*, if there is a boost contract, the connected boost contract will update its *_balances*. If a user withdrew from the vault while these contracts were disconnected, the boost contract will continue to

	generate yields based on the higher *_balances* value. If a user deposited more, they will have to withdraw their stake and redeposit to start earning the correct amount.
RECOMMENDATION	Directly synchronize values between the vault and boost contract. Otherwise, ensure that the boost contract is not disconnected until the boosted rewards are completely distributed.
RESOLUTION	A lock was added to prevent the boost contract from being disconnected while it is distributing rewards. Care should be taken to not extend boost contracts via *notifyRewardAmount()* once they have been disconnected. Reviewed in commit 904efd045788b81838fe3ae4caf73540858d8309

Static Analysis

Missing Zero Checks

FINDING ID	#0023
SEVERITY	Informational
STATUS	Closed
LOCATION	• AbstractVampireStrategyV1.sol -> 32-48: *constructor(//...* • BaseVampStrategy -> 36-50: *constructor(//...* • SingleStakingBoost.sol -> 36-46: *constructor(//...* • StratManager.sol -> 30-42: *constructor(//...* • VampireVaultMultiV3.sol -> 59-63: *constructor(//...* • VampireVaultMultiV3.sol -> 341-346: *function setBoostContract(address _boostCon, bool _boost, uint256 pid) public onlyOwner {* • VampireVaultMultiV3.sol -> 355-357: *function addStrategy(address strategy) public onlyOwner {* • VampireVaultMultiV3.sol -> 365-373: *function proposeStrat(address _implementation) public onlyOwner {* • VampireVaultMultiV3.sol -> 417-426: *function inCaseTokensGetStuck(address _token) external onlyOwner {*

DESCRIPTION	Functions don't check for a zero address before assigning variables. Zero addresses may cause incorrect contract behavior.
RECOMMENDATION	Add a check for zero address.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 8c2053002cedfb9b9176f006658dd57e643657dd

No Events Emitted For Changes To Protocol Values

FINDING ID	#0024
SEVERITY	Informational
STATUS	Closed
LOCATION	• FeeManager.sol -> 19: *function setCallFee(uint256 _fee) external onlyManager {* • FeeManager.sol -> 26: *function setFeeCap(uint256 _fee) external onlyManager {* • FeeManager.sol -> 30: *function setTotalFee(uint256 _fee) external onlyManager {* • StratManager.sol -> 60: *function setKeeper(address _keeper) external onlyManager {* • StratManager.sol -> 68: *function setStrategist(address _strategist) external {* • StratManager.sol -> 77: *function setUnirouter(address _unirouter) external onlyOwner {* • StratManager.sol -> 85: *function setVault(address _vault) external onlyOwner {* • StratManager.sol -> 89: *function setBoostContract(address _boostContract) external onlyOwner {* • StratManager.sol -> 97: *function setravenFeeRecipient(address _ravenFeeRecipient) external onlyOwner {* • AbstractVampireStrategyV1.sol -> 139: *function setBoosted(bool _isBoosted) external onlyVault {* • VampireVaultMultiV3.sol -> 355: *function addStrategy(address strategy) public onlyOwner {*

DESCRIPTION	Functions that change important variables should emit events such that users can monitor protocol value changes.
RECOMMENDATION	Add new events and emit events from the functions that change protocol values.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Duplicate Contract Naming

FINDING ID	#0025
SEVERITY	Informational
STATUS	Closed
LOCATION	• OwnableUpgrade.sol -> 17: *abstract contract Ownable is Context {*

DESCRIPTION	The abstract contract within *OwnableUpgrade.sol* is named *Ownable*. This will clash with *Ownable.sol* as they have duplicate contract names.
RECOMMENDATION	Rename the abstract contract to *OwnableUpgrade*.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Unmatched File And Contract Name

FINDING ID	#0026
SEVERITY	Informational
STATUS	Closed
LOCATION	• BaseVampStrategy.sol -> 15: *contract VampStratV1 is AbstractVampireStrategyV1 {* • VampireVaultMultiV3.sol -> 12: *contract VampVaultMultiV3 is Ownable, ReentrancyGuard*

DESCRIPTION	File and contract names should match.
RECOMMENDATION	Rename the files to match the contract names or vice versa.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Incorrect Or Missing Imports

FINDING ID	#0027
SEVERITY	Informational
STATUS	Closed
LOCATION	• BaseVampStrategy.sol -> 17: *import "./IUniswapRouterETH.sol";* • ERC20.sol

DESCRIPTION	Some solidity compilers are case-sensitive. The contract name *IUniswapRouterETH* should be *IUniswapRouterEth* instead. The *ERC20* contract is missing the following imports: • Context.sol • IERC20.sol • SafeMath.sol • Address.sol
RECOMMENDATION	Correct the case-sensitive import file names and add the missing imports.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit a29b79d120950d5bfb30a9bb665867abe4f5e780

Unused Variables

FINDING ID	#0028
SEVERITY	Informational
STATUS	Closed

LOCATION	• BaseVampStrategy.sol -> 23: *address constant public quick = address(0x831753DD7087CaC61aB5644b308642cc1c33Dc13);*
	• BaseVampStrategy.sol -> 24: *address constant public sushi = address(0x0b3F868E0BE5597D5DB7fEB59E1CADBb0fdDa50a);*
	• BaseVampStrategy.sol -> 25: *address constant public dquick = address(0xf28164A485B0B2C90639E47b0f377b4a438a16B1);*
	• BaseVampStrategy.sol -> 28: *address public sushiRouter = address(0x1b02dA8Cb0d097eB8D57A175b88c7D8b47997506);*
	• BaseVampStrategy.sol -> 29: *address public quickRouter = address(0xa5E0829CaCEd8fFDD4De3c43696c57F7D7A678ff);*
	• BaseVampStrategy.sol -> 32: *address[] public rewardToEth = [reward, eth];*
	• BaseVampStrategy.sol -> 34: *address[] public wantPath2 = [elk, want]; // path to get to cxo*
	• FeeManager.sol -> 12: *uint public FEE_CAP = 10000;*

DESCRIPTION	The above variables are set but never used.
RECOMMENDATION	Remove the variables or incorporate them into the contract functionality.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit 904efd045788b81838fe3ae4caf73540858d8309

On-Chain Analysis

Unverified Contract

FINDING ID	#0039
SEVERITY	High Risk
STATUS	Closed
LOCATION	BaseVampStrategy 0x26439218553e211022B663C469505ecF71989Eb1

DESCRIPTION	The noted contract is unverified. As the strategy will receive tokens from the vault, it is important to verify it on the explorer.
RECOMMENDATION	Verify the contract.
RESOLUTION	The BaseVampStrategy contract was verified.

Privileged Addresses Are EOA

FINDING ID	#0040
SEVERITY	Informational
STATUS	Closed
LOCATION	VampireVaultMultiV3 0x704AAeF878F767B89A1f54d7FDc2d95558107305 BaseVampStrategy 0x26439218553e211022B663C469505ecF71989Eb1

DESCRIPTION	The owner, keeper, and strategist addresses of the vault and strategy are externally owned accounts. Owner functions that can severely impact the contract are restricted by a built-in timelock of 72 hours. Owner/Strategist: 0xab91be9C89Eb7C38b52abd60ce3DE24Ea36a4db0 Keeper: 0x39043f59D85BD60992eD33f54f4A88e08280326B
RECOMMENDATION	No changes are necessary.
RESOLUTION	N/A

Appendix A - Reviewed Documents

Document	Address
AbstractVampireStrategyV1.sol	N/A
Address.sol	N/A
BaseVampStrategy.sol	0x26439218553e211022B663C469505ecF71989Eb1
Context.sol	N/A
ERC20.sol	N/A
FeeManager.sol	N/A
IACVault.sol	N/A
IBoost.sol	N/A
IERC20.sol	N/A
IRewardPool.sol	N/A
IStrategy.sol	N/A
IUniswapRouterETH.sol	N/A
IUniswapV2Pair.sol	N/A
IVampStrategy.sol	N/A
IVault.sol	N/A
Math.sol	N/A
Ownable.sol	N/A
OwnableUpgrade.sol	N/A
Pausable.sol	N/A
ReentrancyGuard.sol	N/A
SafeERC20.sol	N/A
SafeMath.sol	N/A
SingleStakingBoost.sol	N/A
StratManager.sol	N/A

VampireVaultMultiV3.sol	0x704AAeF878F767B89A1f54d7FDc2d95558107305

Revisions

Revision 1	eef9cd2c9ce032a128653c045199e9ed461707f4
Revision 2	a29b79d120950d5bfb30a9bb665867abe4f5e780
Revision 3	8c2053002cedfb9b9176f006658dd57e643657dd
Revision 4	937e4131e9e50e6e11a2ecf395f551223496db15
Revision 5	22eb40151fb765ca7894422fdca60566a9708cd3
Revision 6	1f39282b1a9652ee0c5db9f2386c8aa5525ae5b6
Revision 7	904efd045788b81838fe3ae4caf73540858d8309

Imported Contracts

OpenZeppelin	3.2.0 (files copied into repository)

Externally Owned Accounts

owner/strategist	0xab91be9C89Eb7C38b52abd60ce3DE24Ea36a4db0
keeper	0x39043f59D85BD60992eD33f54f4A88e08280326B
ravenFeeRecipient	0xCFd5b467e74A51cdc55e89Ac5E33BFFc3DB46830

External Contracts

These contracts are not part of the audit scope.

Vault Reward Token Strategy Want Token	RavenVaultILPV1 (ravenELKSS) 0x640Ee5105B01b612668b599A879da3E230A8d0FE
Strategy LP Token	Elk Liquidity Pair (WMATIC/ELK) 0x7Cb0703aa37601a02798BDFF63A18aF2dD082572
Elk Token	0xE1C110E1B1b4A1deD0cAf3E42BfBdbB7b5d7cE1C
Elk Router	0xf38a7A7Ac2D745E2204c13F824c00139DF831FFf
WMATIC	0x0d500B1d8E8eF31E21C99d1Db9A6444d3ADf1270

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.

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

- deployed contracts match those which were audited in manual/static analysis;
- contract values are set to reasonable values;
- contracts are connected so that interdependent 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