Vesper Pools

February 2022

coinspect

Vesper Pools

Smart Contract Audit

V220421 Prepared for Vesper • February 2022

1. Executive Summary

2. Assessment and Scope

3. Summary of Findings

4. Detailed Findings

   VSP-46 Loss of funds through low priced swaps

   VSP-47 _getRewardAccrued reverts when COMPTROLLER is uninitialized

   VSP-48 Overestimated rewards

   VSP-49 Updating rewardTokens could miss unclaimed rewards

5. Disclaimer

1. Executive Summary

In **February 2022, Vesper** engaged Coinspect to perform a source code review of **Vesper Pools**. The objective of the project was to continue to evaluate the security of the smart contracts.

This audit focused on the following updates: adding support for the Avalanche blockchain, support for two Compound forks that run on Avalanche, changes in the `Convex` and `Curve` strategies, and pool code updates.

In **April 2022** Coinspect conducted a new review limited to the code modified to address the security issues identified during the initial assessment.

The following issues were identified during the initial assessment and their status has been updated on the latest review:

High Risk	Medium Risk	Low Risk
0	**2**	**2**
Fixed 0	Fixed 0	Fixed 2

Issues `VSP-47` and `VSP-49` were introduced by the code refactoring of strategies. Issue `VSP-46` is caused by the lack of an oracle price reference in the new Avalanche strategy. `VSP-48` arises from ignoring side effects when simulating calls inside a view function.

2. Assessment and Scope

The audit started on **February 7, 2022** and was conducted on the Git repository at https://github.com/bloqpriv/vesper-pools-v3. The commit reviewed during this engagement was `9a484017e5adc63b428bdbf0565352ea1e90d328` from **February 7, 2022**.

The scope of the audit was limited to the latest version of the following Solidity source files, shown here with their sha256sum hash:

```
81674c559f4ee1a8c439229fd7726272014c8746b94724596dffae2e7a2a7123  interfaces/curve/ILiquidityGauge.sol
4ef4dfae45363ae3c61ec9db619576eed0e58229a53ae0ef47995772ce59c885  pool/PoolAccountant.sol
dd70d0c32b0b4f92cebc8db6e7612ad21ee7fd4398a2ac85c21e637a2e3f502f  pool/PoolRewards.sol
22995d31ff8a29aea1b03efd450400d566033747dc9905e41f6bd020d1be153c  pool/VPool.sol
a3d09800178df6282ec3fc699af72d91954b9ffa4c65f94b9b3b26168cdd1215  pool/VPoolBase.sol
ae95ba4f98b764edccd7651bb5f75e1bc3089ebd7c1c66115102b95936a2afb4  pool/vfr/VFRStablePool.sol
bf3c3b333cdf7ce4dcd4c16473ec26aa87649c950574609636b9a6615c257e3e  strategies/Earn.sol
8bd3265e8ddc81c16bf9073865663a748efc85827e0c171160462027c476d999  strategies/Strategy.sol
f8e276ad776a777aebe95fa21be4d38ea9c2f0c14b51604b9fcdf484ae898e5f  strategies/aave/AaveStrategy.sol
8004a3c19000e40826bd86cc7d35e0b69f3919904698da8e577519138b7edd6a  strategies/aave/AaveStrategyAvalanche.sol
bb135ee417828efdf5d9abfdb3b72fe138f2044fae62a605afce48350edb7e4e  strategies/aave/AaveStrategyPolygon.sol
452ae8cd3548e3b511ebb2acae4cf8216a244c3019f2f1eee06eaa03fc9c3916  strategies/aave/AaveV1Strategy.sol
b03c251d9956571e74974f58c0beb024ff19b94b99a22eacffda05c03fdc1648  strategies/alpha/AlphaLendStrategy.sol
1bd224262117c13dbe759b62e8d2eca7c88c80879563175d3393fd726af6c8b8  strategies/compound/CompoundLeverageStrategyETH.sol
ca95d9169a9d13649bb7ac2ae638ee9c982fd122e3def7ddf1a3283a90129369  strategies/compound/CompoundLeverageStrategyLINK.sol
09996580f5405891ccdb5571ebec75586632af5734defe1f29377b413bbfa694  strategies/compound/CompoundLeverageStrategyUNI.sol
638b429a5bcd6817ccb56d279ec1f9de2b657193f2ad7c75c6a13a1da7daf871  strategies/compound/CompoundStrategy.sol
a0d2a3fb095bc2ddf8d5d35a7ecb04820d8e2c1ae39f1de520c4c3f32b335418  strategies/compound/CompoundStrategyETH.sol
b6bedcab172dec7f3c99308f5c80ca5c3279001a8726a8ec1a76401811e71d55  strategies/compound/CompoundXYStrategyETH.sol
e39e3f34a99e23de87296af66956fa513985cf1b680ca92b10ffa24647cdd93d  strategies/compound/earn/EarnCompoundStrategy.sol
48fcf7d0bb8c6b880e800257404feea5ad9fd9bc3678b0394e1ea57cb52b5305  strategies/compound/earn/EarnCompoundStrategyETH.sol
a3412fefc95053a54f8820503bb8c0527c4c82dc833d19019e61ceaddbffa533  strategies/compound/vfr/CompoundCoverageStrategy.sol
cc315d2573accba6362af44ed22713c7514e41f99bc8a14cee4e3a864499e9cb  strategies/compound/vfr/CompoundStableStrategy.sol
03c26250de5d2a781752c891aa3dba26a3209a9c30199c2a08a07d54f1b38885  strategies/convex/2Pool/Convex2PoolStrategy.sol
e22bfc0e1c8d3f7243c4db2c9f4141344fd1fcbe7f7b1ae9af12f3e6f4163f1e  strategies/convex/2Pool/Convex2PoolStrategyMIMUSTPool.sol
897d6fe884f89dcfc75c8dea5f949fca2dd544eb050c4ce4f7c2cbea9b5aaa32  strategies/convex/4Pool/Convex4MetaPoolStrategy.sol
f42103d786dd2f5ee759b220d1db408de392dd4b3d65fb021a271053c8c8aaa7  strategies/convex/4Pool/Convex4MetaPoolStrategyMIMPool.sol
6582eaa0e7e523cf362b355c23d86a4a1c9cb2b12207bf28c6ee7cd1e5426af0  strategies/convex/4Pool/Convex4PoolStrategy.sol
76b0c39a6a0adc0bc0c1e886376dead7fcf811519825eaada064c473fd0dca5d  strategies/convex/4Pool/Convex4PoolStrategySUSDPool.sol
59011cc7e6d2349d47fa603b4e638affdafa54897fe4ad522bdf8ba49e7f2263  strategies/convex/ConvexStrategy.sol
3b3c616eec0ccdcb21638553cf9ecbb1b19ce9c4542390aacd29cc62f6210579  strategies/convex/ConvexStrategyBase.sol
c1f72cef10e6a3981d08304741c4c564b896eee2af21c2f41cdb1b4107e9ef6a  strategies/convex/vfr/ConvexCoverage3PoolStrategy.sol
506ef88b49adaa1abc53f2a30c1d6026ed6ec01bc078dc4a7944b9ab0308e96e  strategies/convex/vfr/ConvexCoverageStrategy.sol
fb1f646cdbc0abe10212a4c28cb02feb99e575a7d6fd339d789a2e647df12ea8  strategies/convex/vfr/ConvexStableStrategy.sol
6e7bf7ac0f8723bf063fe570e3b79b99f8eca8b03dc9b52e4aa6c94fdd3a726a  strategies/curve/2Pool/Crv2PoolStrategy.sol
e1827cd8df192e23cc304821bb80e70c2f445c02d8918b5ff77b27c2c18c597a  strategies/curve/2Pool/Crv2PoolStrategyArbitrumUSDCUSDTPool.sol
3b0d3430b6f093b0351ee5a7e9168e32674c90e4e5ea5b52182d7a20cb32818b  strategies/curve/3Pool/CrvSBTCPoolStrategy.sol
42138d2e6428b1ee0b319c5869773590cc7394603cf5c8f10f08d77f45a7b1fa  strategies/curve/3Pool/earn/EarnCrvSBTCPoolStrategy.sol
5b2dd5afee177b4e29e5389f21b853570a888dc58b8d1a0fc8bd7f8f34af36ce  strategies/curve/4Pool/Crv4MetaPoolStrategy.sol
c45e656697e04734231274a89edb1d39012d0291f11af0e837de704f7255b950  strategies/curve/4Pool/Crv4PoolStrategy.sol
005d45dfe285e5bb50065e53b7dfc2141293edebdaa905736133ba9c44397f1a  strategies/curve/CrvBase.sol
f5ca596f1a9533b078bcf66c346c53a4ff6419b215022c8e8ca0acb5d764ccac  strategies/curve/CrvPoolStrategyBase.sol
005ea0ccbf5540a63703d2ed1f98ba96aff9b3273e6b4d5276a9c76711a37f8e  strategies/curve/a3Pool/CrvA3PoolStrategy.sol
c29cc054605638a8808d627b1c3b0d22b0c1f3882e3cbb2412229e16794911f0  strategies/maker/MakerStrategy.sol
2bbe50b9d307e5a50ff756eb88eded9c76e67f56f20a350889c5b46a04b54b21  strategies/rari-fuse/RariFuseStrategy.sol
cb37bb219b91dc6b41edd38eaa123f8cac6dfe95ed581838a1ecee61a732d835  strategies/rari-fuse/earn/EarnRariFuseStrategy.sol
f692321bd21a8d26825ad5970aec345bb82ea84c10ee3f9a78693bf2da692c99  strategies/vesper/VesperStrategy.sol
bb72a9933eb707999bbf91890b6f745e69cdd13491323846452e99ead9d4aff5  strategies/yearn/YearnStrategy.sol
561cfacc91a799ce691fd7df88e50419ee164c0e936f033fd4d5f1cd616e4cde  upgraders/VPoolUpgrader.sol
```

The contracts are specified to compile using **Solidity 0.8.3**. It is recommended to update to Solidity compiler version **0.8.4** because this version fixed an important

bug. See Solidity ABI Decoder Bug For Multi-Dimensional Memory Arrays for more information.

Avalanche strategies

A new strategy for the Avalanche blockchain was added.

The strategy invests in a fork of Compound using the `WAVAX` token instead of `WETH`. Because of this, the strategy inherits from the `CompoundStrategy`. The DEX platforms used for investments are Benq and TraderJoe. TraderJoe adds a `RewardDistributor` contract. This contract accumulates rewards from the platform that must be claimed through it. Then, the reward distributor contract pays the strategy the awarded value.

Coinspect found one issue in this strategy related to the lack of utilization of an oracle for price references when swapping tokens. This could lead to issues where the awarded amount in the collateral token is lower than it should be. See `VSP-46`.

Curve and Convex strategies

The Curve and Convex strategies were modified to support different Compound forks. This required a simple refactor and the creation of the `ConvexStrategy` contract.

New strategies were added: `ConvexD3PoolStrategy` (based on the new `ConvexStrategy` contract), `Convex4MetaPoolStrategyIBBTCPool`, `Convex4MetaPoolStrategyFRAXPool` and `Convex4PoolStrategyMUSDPool`.

Other changes

Besides the aforementioned changes, there are other modifications spread over the code base that were also reviewed:

- Supporting no `growPool` in Earn strategies
- Governor can call `onlyKeeper` methods

- Strategies catch failed oracle calls reverts
- `Comptroller` is set at deploy time and can be null. However, the rest of the existing code is not prepared to handle a zero value in the `comptroller`, requiring an adaptation of the code or forbidding this value. See `VSP-47`.
- Curve strategies can update `rewardTokens` when the underlying Curve contract does so. This created a simple border case where updating the `rewardTokens` in some cases can lead to loss of funds. See `VSP-49`.

3. Summary of Findings

Id	Title	Total Risk	Fixed
VSP-46	Loss of funds through low priced swaps	Medium	✘
VSP-47	_getRewardAccrued reverts when COMPTROLLER is uninitialized	Medium	✘
VSP-48	Overestimated rewards	Low	✔
VSP-49	Updating rewardTokens could miss unclaimed rewards	Low	✔

4. Detailed Findings

VSP-46 Loss of funds through low priced swaps

Total Risk	Impact	Location
Medium	Medium	`CompoundMultiRewardAvalancheStrategy.sol`
Fixed ✘	Likelihood Medium	

Description

The strategy could be forced into unprofitable token swaps.

When claiming the `rewardToken` in the `CompoundMultiRewardAvalancheStrategy` strategy, the code uses no price reference for swapping tokens. The `_claimRewardsAndConvertTo` function of the `CompoundMultiRewardAvalancheStrategy` does not use any price reference for swapping the reward token. Instead, the minimum expected amount of output tokens is set to 1, bypassing this protection mechanism.

```
61  function _claimRewardsAndConvertTo(address _toToken) internal virtual override {
62      ComptrollerMultiReward(address(COMPTROLLER)).claimReward(0, address(this)); /
Claim protocol rewards
63      ComptrollerMultiReward(address(COMPTROLLER)).claimReward(1, address(this)); /
Claim native AVAX (optional)
64
65      uint256 _rewardAmount = IERC20(rewardToken).balanceOf(address(this));
66      if (_rewardAmount != 0) {
67          _safeSwap(rewardToken, _toToken, _rewardAmount, 1);
68      }
69      uint256 _avaxRewardAmount = address(this).balance;
70      if (_avaxRewardAmount != 0) {
71          TokenLike(WAVAX).deposit{value: _avaxRewardAmount}();
72          if (_toToken != WAVAX) {
73              _safeSwap(WAVAX, _toToken, _avaxRewardAmount, 1);
74          }
75      }
76  }
```

Accepting an arbitrary price for a token exchange can result in undesired loss of funds.

Recommendation

Use a price oracle to calculate the minimum expected amount for the last `_safeSwap` parameter.

Status

April 20, 2022: After testing, the Vesper team concluded that saving gas was more impactful than accounting for slippage as earnings between rebalances are low values.

VSP-47	_getRewardAccrued reverts when COMPTROLLER is uninitialized	
Total Risk **Medium**	Impact Medium	Location `CompoundStrategy.sol`
Fixed ✘	Likelihood Medium	

Description

An unhandled call to a possible `address(0)` occurs in the internal `_getRewardAccrued` function of the `CompoundStrategy` which is called from the `totalValue` function.

```
89  function _getRewardAccrued() internal view virtual returns (uint256 _rewardAccrue
90      _rewardAccrued = COMPTROLLER.compAccrued(address(this));
91  }
```

The `COMPTROLLER` variable may be zero as stated in the constructor:

```
        // Either can be address(0), for example in Rari Strategy
        COMPTROLLER = Comptroller(_comptroller);
        rewardToken = _rewardToken;
```

This can cause unexpected reverts when calling the `_getRewardAccrued` function.

A similar issue appears in the `_claimRewards` function.

Recommendation

The `CompoundStrategy` should handle the case where the `COMPTROLLER` variable is zero or forbid such a value.

Status

April 20, 2022: The Vesper team considers that the comptroller will never be 0. Accepting `address(0)` at construction time allows for fetching the value dynamically.

VSP-48	Overestimated rewards		
Total Risk **Low**	Impact Low	Location `Convex2PoolStrategy.sol`	
Fixed ✔	Likelihood Low		

Descriptions

In Convex strategies the rewards could be overestimated.

This happens due to the `view` nature of the `claimableRewardsInCollateral` function as it is used.

```
77  function claimableRewardsInCollateral() public view virtual override returns (uint256 rewardAsCollateral
78      ClaimableRewardInfo[] memory _claimableRewardsInfo = _claimableRewards();
79      for (uint256 i = 0; i < _claimableRewardsInfo.length; i++) {
80          if (_claimableRewardsInfo[i].amount != 0) {
81              (, uint256 _reward, ) =
82                  swapManager.bestOutputFixedInput(
83                      _claimableRewardsInfo[i].token,
84                      address(collateralToken),
85                      _claimableRewardsInfo[i].amount
86                  );
87              rewardAsCollateral += _reward;
88          }
89      }
90  }
```

This function estimates the rewards in a `for` loop that does not take into account the effects of actually swapping the same tokens. When swapping the values in a similar loop, the initial swaps may affect the value of the collateral token, actually reducing the amount received.

This function is currently unused by the smart contracts, but an external process or a future smart contract may rely on it with unexpected consequences.

Recommendation

Document this behavior to avoid future errors. Consider renaming the function to `estimateClaimableRewardsInCollateral`, for example.

Status

April 20, 2022: The function has been renamed in commit `58393e12acc9c69aeddbfa6201c0357b81c1b563`.

VSP-49	Updating rewardTokens could miss unclaimed rewards		
Total Risk Low	Impact Low	Location ConvexStrategy.sol, Convex4PoolStrategy.sol, Convex4MetaPoolStrategy.sol, Convex2PoolStrategy.sol	
Fixed ✔	Likelihood Low		

Description

The `setRewardTokens` function in Convex strategies could miss unclaimed rewards if tokens are removed.

An example case:

```
28  function setRewardTokens(
29          address[] memory /*_rewardTokens*/
30          ) external override onlyKeeper {
31      rewardTokens = _getRewardTokens();
32      _approveToken(0);
33      _approveToken(MAX_UINT_VALUE);
34      _setupOracles();
35  }
```

The new token list can have either more or less tokens than the original.

In any case, old tokens are still approved and could hold some value.

Recommendation

Make a call `_approveToken(0)` before calling `_getRewardTokens` and create a mechanism that claims unclaimed values if it is necessary (this can be guaranteed by the keepers through off-chain software).

Status

April 20, 2022: Rewards are claimed before updating the `rewardTokens`. The fix was added in commit `b4d0ef85c32e140035b998575eb23184f1a15041`.

5. Disclaimer

The information presented in this document is provided "as is" and without warranty. The present security audit does not cover any off-chain systems or frontends that communicate with the contracts, nor the general operational security of the organization that developed the code.