Smart Contract Audit

Vesper - March 2022

Vesper Pools

Smart Contract Audit

V220311

Prepared for Vesper • March 2022

1. Executive Summary

In **March 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: refactoring required to add support for Compound forks deployed on multiple chains, several new strategies and an update to the RariFuseStrategy rewards claiming mechanism.

The following issues were identified during the assessment:

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

Issue `VSP-50 and VSP-51` are caused by the lack of an oracle price reference when swapping tokens in strategies.

2. Assessment and Scope

The audit started on **March 7, 2022** and was conducted on the Git repository at https://github.com/bloqpriv/vesper-pools-v3. The commit reviewed during this engagement was `bf3bfbfd911a588d43e37ecee7bcadfa3e66c7ca` from March 3, 2022.

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

```
e2593ad2942f98da9ac498b1de5787f90b080d2ee65a6e8fb409255de5af2758  FlashLoanHelper.sol
fb97a0a096024929ed67f3ff0216ea66fff499d5eaba909a082849df027196a7  interfaces/compound/IComptrollerMultiReward.sol
9247e11358e933a73b3d15560195bb713a19ed8e7425a2fed38b643a216ab014  interfaces/rari-fuse/IComptroller.sol
7ad32030dfeba7497dc85c69174f127e155814406520d558b5671036d56dd231  strategies/compound/BenqiCompoundLeverageAvalancheStrategyAVAX.sol
a03fc3142aeae4bb22f1e46ca72436551b6c3831684f7fbb595078a3c3fdd88a  strategies/compound/BenqiCompoundMultiRewardAvalancheStrategyAVAX.sol
b971a5e58e9f876130141b16020cdb68aef769c6bb24a366d12ff64423402420  strategies/compound/CompoundLeverageAvalancheStrategy.sol
4087877e3824bc3ec3c130a04c3a6f6c3d215215de6f2290b09fb2edaad00752  strategies/compound/CompoundLeverageStrategy.sol
d25a57c23d8b9823687c8446e0a93a4389d2ba16b168ca644a2083dc209abe33  strategies/compound/CompoundLeverageStrategyETH.sol
a5db991a0393e1cce5ee2e560bdf56e563848afc0dd44a5f3fe9e7da8ccdbb5e  strategies/compound/CompoundMultiRewardAvalancheStrategy.sol
9ac1c48dda96404d2bdc6f224da170483b5470bb93b4a7d5ad3272c857ff0822  strategies/compound/CompoundStrategy.sol
1d33122374c1111e3c0a6c9556d1ad7fd1b8d2beae65cadd2bdd7a4c9b8132e8  strategies/compound/CompoundXYStrategy.sol
e89795000ff353ffeb746bea1ebe263267a71a8b6bfe5c9202dd58129e6fa410  strategies/compound/CompoundXYStrategyETH.sol
8a5060a94ef4635a2e89e22fa35efec6a4ddf48b9e0835a5ff5295a644e172b0  strategies/compound/VesperCompoundXYStrategy.sol
172e1f80968a30cad478a899964ba72a553723e5b237fe5b5f8a691f8916b59d  strategies/rari-fuse/RariCore.sol
33a80856cf9044608e05ac364c86edb91c2de51ef6937d2eeb13bcb780d79da6  strategies/rari-fuse/RariFuseLeverageStrategy.sol
00a2509355b2aac942fa024813fde3ce78e1c408d05a43e0fbbfc44c96b065c6  strategies/rari-fuse/RariFuseStrategy.sol
097205d171ff1b4ef578e77cd95400b49445ed88c81213c25f5f82516c18c741  strategies/rari-fuse/RariFuseStrategyETH.sol
87145bfd6974d9ee11cb6526029910a968f04557f9de5860010234090cd635ad  strategies/rari-fuse/earn/EarnRariFuseStrategy.sol
ede53018493801ba6ffb865c23ca5a40902afdfbedd31113ff658d8d2f88d733  strategies/rari-fuse/earn/EarnRariFuseStrategyETH.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.

This engagement concentrated on the changes introduced since Coinspect's previous audit. The most important changes introduced included:

1. The Rari Fuse strategies were updated to claim rewards in fuse pools.
2. The Compound strategy was refactored to enable the development of new strategies for Compound like forks in other chains, such as the Benqi strategies for the Avalanche chain. For this purpose, three hooks were added: `_afterBorrowY`, `_beforeRepayY` and `_rebalanceBorrow`.
3. The following new strategies were added:
    a. `VesperCompoundXYStrategy`
    b. `CompoundLeverageAvalancheStrategy`
    c. `BenqiCompoundLeverageAvalancheStrategyAVAX`
    d. `CompoundLeverageAvalancheStrategy`
    e. `BenqiCompoundMultiRewardAvalancheStrategyAVAX`

    f. `RariFuseLeverageStrategy`
    g. `CompoundMultiRewardAvalancheStrategy`

The following addresses referencing external (and Vesper's VSP) contracts are used and were verified to be correct:

1. `0xB31f66AA3C1e785363F0875A1B74E27b85FD66c7` for `WAVAX` as listed in https://support.avax.network/en/articles/5232004-what-is-the-avalanche-avax-token-s-contract-address
2. `0x4Ddc2D193948926D02f9B1fE9e1daa0718270ED5` for `CETH` as listed in https://compound.finance/docs
3. `0x1b40183EFB4Dd766f11bDa7A7c3AD8982e998421` for `VSP` as listed in https://docs.vesper.finance/vesper-grow-pools/vesper-grow/audits

3. Summary of Findings

Id	Title	Total Risk	Fixed
VSP-50	Loss of funds through low priced swaps	Medium	✘
VSP-51	Loss of funds through low priced swaps in Avalanche	Medium	✘
VSP-52	_getCollateralFactor magic number	Info	✘

4. Detailed Findings

VSP-50 Loss of funds through low priced swaps

Total Risk	Impact	Location
Medium	Medium	`VesperCompoundXYStrategy.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 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:

```
    _safeSwap(VSP, _toToken, _vspAmount, 1);
```

Also, the `_rebalanceBorrow` function is implemented in a similar fashion:

```
    _safeSwap(borrowToken, address(collateralToken), _borrowedHere, 1);
```

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

Recommendation

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

VSP-51 Loss of funds through low priced swaps in Avalanche

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

Description

Avalanche strategies disable slippage protection for token swaps and could be forced into unprofitable token swaps.

The overridden `_safeSwap` function 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:

```
function _safeSwap(
    address _tokenIn,
    address _tokenOut,
    uint256 _amountIn
) internal override {
    // Removed UniV3 Oracle slippage check on Avalanche
    _safeSwap(_tokenIn, _tokenOut, _amountIn, 1);
}
```

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

Recommendation

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

VSP-52 _getCollateralFactor magic number

Total Risk	Impact	Location
Info	-	`RariFuseLeverageStrategy.sol` `CompoundLeverageStrategy.sol`

Fixed	Likelihood
✘	-

Description

The `_getCollateralFactor` function implemented in 2 different strategies adjusts the `collateralFactor` using a hard-coded number.

```
// Take 95% of collateralFactor to avoid any rounding issue.
_collateralFactor = (_collateralFactor * 95) / 100;
```

Recommendation

Define the 95% magic number as a constant in order to make code more maintainable and avoid potential mistakes in the future.

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.