Smart Contract Audit
Vesper Strategies

Prepared for Bloq • March 2021

v210326

1. Executive Summary

In March 2021, Bloq engaged Coinspect to perform a source code review of two new investment strategies added to the Vesper platform. The objective of the audit was to continue evaluating the security of the smart contracts being developed.

The assessment was conducted on contracts from the Git repository at https://github.com/bloqpriv/bpools obtained on **March 18th**.

Overall, Coinspect did not find any high risk security vulnerabilities introduced by the two new strategies or the modifications made to the Vesper since its previous audit that would result in stolen or lost user funds.

The following issues were identified during the assessment:

High Risk	Medium Risk	Low Risk
0	0	1

The only low risk finding reported details how gas utilization can be optimized by changing the visibility of two functions to external in a new contract not yet deployed (See unsafeGetAmountsOut/In should be declared external to save gas).

The following report details the tasks performed and the vulnerabilities found during this audit as well as several suggestions aimed at improving the overall code quality and warnings regarding potential issues.

2. Introduction

The focus of this audit were several changes performed to existing contracts and the recently added Vesper-Maker and Compound-Maker strategies. The changes evaluated consisted in refactoring of strategies, gas optimizations and bug fixes.

The audit started on March 22th and was conducted on the Git repository at https://github.com/bloqpriv/bpools. The last commit reviewed during this engagement was `e7821dc95469ca87dd58458411bf8eaf349cf5fb` from **January 6th**:

```
commit e7821dc95469ca87dd58458411bf8eaf349cf5fb
Merge: d4a1b29 8895bd6
Author: patidarmanoj10 <32006767+patidarmanoj10@users.noreply.github.com>
Date:   Thu Mar 18 10:46:04 2021 -0700

   Merge pull request #477 from bloqpriv/strategy-migration

   Added logic to support strategy migration
```

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

```
c3b96684a68f6eb9b8b81e6299fee4c9b08a1476ab1701652dfcafa519608ffd  ./strategies/AaveMakerStrategy.sol
d816a99bf5d2ca5e98fc93c48ae65666d497d0c9aeef8c2615a6d9b27e2b8781  ./strategies/AaveV2Strategy.sol
9c962bd8eabc7a9c11102d02fbc4abec950f828077a6220cb4f31cfe785ebfa7  ./strategies/AaveV2MakerStrategy.sol
d9f28f55450f152c812410e048255e9bc0c3c59f301ef34310d177b9fcdd513c  ./strategies/CompoundStrategy.sol
1ebf288152fc0d0fefe6a137222531931e94401669456df848ae0560e20cbf0a
./strategies/CompoundMakerStrategyETH.sol
ea3f9c5a66254ea9ff731b9b3d04d893d690ecc7d9f43344ff4436ca4e0851d5
./strategies/CompoundMakerStrategy.sol
090e8563554cf4c3975b5faa564b9688fbe5cdc5d8268bad157686d5b18b672e  ./strategies/AaveStrategy.sol
dc8a5bd1fb50996654edef5c04d14ca4e5188ef2c783d6ac7eeb43d6f36753af  ./strategies/MakerStrategy.sol
bd33e7bc61311bc2db66191e98d7e16d62ddc6e2e569b83a0305a6263cd3d95b  ./strategies/Strategy.sol
e22efab606ea41ac51e39abfa5ec5c6b2a75165903b3059612ce9dc12474bfc7  ./strategies/VSPStrategy.sol
403c015dd4099721bd02f55e8b8f1109ea40ac7166c8ed9ca50ae6715dcc29a4  ./strategies/VesperMakerStrategy.sol
495ab4a192b0dbdb8df0e65753e4c2b37ae13e7c759981ca5343627c5d0e70f7
./strategies/VesperMakerStrategyETH.sol
a6f2345bc361dfb2b094f43bf5a8d32cb9a7ea83ddbe342c8fd0cb25be5de906  ./strategies/CollateralManager.sol
dc3b34f5b3792ad51876213a3db77801b8575a0bff45af3017481e9b211b4291  ./UniswapManager.sol
```

These smart contracts interact with multiple other contracts deployed by other projects which were not part of this review such as: MakerDAO, Uniswap, Compound and AAVE. A full review comprising these interactions should be performed in order to fully assess the security of the project.

3. Assessment

Vesper is a DeFi ecosystem and growth engine for crypto assets. It provides a suite of yield-generating products, focused on accessibility, optimization, and longevity.

The platform implements crypto asset pools that enable users to generate earnings using different strategies that supply liquidity to existing 3rd party pools in the DeFi ecosystem.

The code reviewed is currently under active development and consists of several pools and strategies which handle different types of collateral deposits and invest them in different projects.

This incremental audit performed as per Vesper request focused on two new investment strategies soon to be deployed and a set of changes recently introduced to some of the contracts already audited in previous engagements, in order to establish if these modifications affected in any way the platform security.

Most contracts are compiled with Solidity compiler version 0.6.12 and it is advised to migrate to a newer version whenever possible as this will increase protection from unknown vulnerabilities. For a recent example see this vulnerability fixed in the compiler https://blog.soliditylang.org/2021/03/23/keccak-optimizer-bug a few days ago. Coinspect verified Vesper is not affected by this issue. Also using a newer compiler would result in gas usage optimizations, for example by using checked arithmetic and not requiring SafeMath.

Coinspect observed the project's code maturity is improving each review and that more tests are continuously being added for each new feature. However, because of the way the tests are structured (they need to be run separately) it has not been possible to obtain a complete code coverage report and it is advised this is improved in the future.

The following sections detail each of the code changes introduced by the commits reviewed by Coinspect.

3.1 New strategies

The CompoundMakerStrategy is a new strategy which supplies borrowed DAI to Compound. It is intended to replace the existing AaveMakerStrategy.

The VesperMakerStrategy is a new strategy which supplies assets to a Vesper pool itself. DAI borrowed from Maker is supplied to Vesper's vDAI pool.

Coinspect observed a minor typo in a comment in VesperMakerStrategy.sol (CMOP should be COMP):

```
* we are using Uniswap to get collateral amount for earned CMOP and DAI.
```

3.3 UniswapManager

This new contract will be used in the future by the rest of the contracts and includes many improvements over how the Uniswap interaction is currently being handled by duplicated code in each strategy and pool.
Helper functions are provided that find the best path for a swap between tokens, trying direct swap when possible or passing through WETH when needed. Also, wrappers that catch reverts in Uniswap and return 0 are included, in order to allow the code to continue executing in those scenarios (e.g., low liquidity pools).

Coinspect verified the following addresses used by the contract are correct:

```
address public constant WETH = 0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2;
IUniswapV2Factory public constant FACTORY =
    IUniswapV2Factory(0x5C69bEe701ef814a2B6a3EDD4B1652CB9cc5aA6f);
IUniswapV2Router02 public constant ROUTER =
    IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D)
```

But it should be noted these could change in the future.

The following commits implement this feature and were reviewed:

- `cd537a2c2e3ee96e98ba65daecff7de677cad43b`
- `a201a89f871fb012ed7074fd49373e855cfe456d`
- `11ec98a2028b4676a9fbb8da1d6418615174044c`
- `9bec61be8a702eb71d75a8eb5df4c60682cfc248`
- `9ffe4cc7bd8091de778d9b0f2a81e87ebee13980`
- `90bf4b4c5c204497fb03c8e9060a9863015208fe`

3.4 VSPStrategy

Several changes were introduced to the existing VSPStrategy:

1. Fixed a rebalance and sweep issue when interacting with Uniswap.
2. The strategy is now pausable by the Controller.
3. The pools list which was previously immutable, now can be updated by the Controller.
4. Added a liquidation limit for each pool, no more than the limit amount is rebalanced each call
5. The strategy no longer pays founder fees.
6. Added a list of keepers that is managed by the Controller. A related `onlyKeeper` modifier was created. The `rebalance` function, which before could be called by anyone, is restricted to the keeper role now.

The following commits implement these changes and were reviewed:

- `4987d4a22f1064448e40728e052eaac134fb4706`
- `91b4f7a4ef1ef95c2d03c51cdd7f0ecd8db5ba15`

3.5 AaveMakerStrategy and CollateralManager

Handling of Maker's DAI minting limit was improved in `AaveMakerStrategy` and `AaveV2MakerStrategy`. The actual DAI balance of the strategy is used instead of the amount passed to the borrow function in the collateral manager.
Also, the `CollateralManager borrow` function was modified to never borrow more than 99% of currently available according to the current debt and vault's limit.

`AaveMakerStrategy` and `AaveV2MakerStrategy` were improved by adding a check before Uniswap swaps to make sure the call does not fail during the rebalance and resurface operations.

The following commits implement these changes and were reviewed:

- `a2e665ef27a6d87feac05061b1d85bba54d50bd9`
- `2d644a0d6ab6070b29c9a4e30d893db7353dbae1`
- `771ad5413d82434f44dacdd498e2649f14c550d8`

3.6 Support for withdraw N via controller

The new modifier `onlyAuthorized` was added. All strategies were modified to replace `onlyPool` with `onlyAuthorized` in all `withdraw` functions. As a consequence, now the controller has the ability to withdraw from the strategies, in addition to the pool that was the only authorized for this purpose before:

```
-    function withdraw(uint256 _amount) external override onlyPool {
+    function withdraw(uint256 _amount) external override onlyAuthorized {
```

The following commit implement this change and was reviewed:

- `80a416080811787e3bf4ba507851d8cf4711f627`

4. Summary of Findings

ID	Description	Risk	Fixed
VSP-021	unsafeGetAmountsOut/In should be declared external to save gas	Low	✘

5. Findings

VSP-021	unsafeGetAmountsOut/In should be declared external to save gas

Total Risk	Impact	Location
Low	Low	UniswapManager.sol
Fixed	**Likelihood**	
✘	Low	

Description

The `unsafeGetAmountsOut` and `unsafeGetAmountsIn` functions in `UniswapManager.sol` should be declared external instead of public in order to save gas:

```
function unsafeGetAmountsOut(uint256 _amountIn, address[] memory path)
    public
    view
    returns (uint256[] memory result)


function unsafeGetAmountsIn(uint256 _amountOut, address[] memory path)
    public
    view
    returns (uint256[] memory result)
```

Those functions that are never called by the contract itself can be declared external in order to optimize gas usage because of the way EVM handles function call parameters. Solidity copies array arguments to memory, external functions read the parameters from the calldata instead; and memory allocation is expensive in comparison to reading calldata.

Recommendation

Review each function declaration and declare as external those that do not need to be public.

6. Disclaimer

The information presented in this document is provided "as is" and without warranty. Source code reviews are a “point in time” analysis and as such it is possible that something in the code could have changed since the tasks reflected in this report were executed. This report should not be considered a perfect representation of the risks threatening the analyzed system.