Smart Contract Audit
Vesper V3

Prepared for Bloq • May 2021

Second audit v210518

1. Executive Summary

In May 2021, Bloq engaged Coinspect to continue reviewing their new implementation of their Vesper platform: Vesper V3. 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/vesper-pools-v3, the last commit reviewed was obtained on **May 17**.

The changes evaluated included the CRV 3Pool strategies ported from V2, new SwapManager, the integration of Uniswap TWAP Oracles, new Maintainers role, bug fixes, and several code refactorings.

Overall, Coinspect did not find any high risk security vulnerabilities introduced by the modifications made to the Vesper since its previous audit that would result in stolen or lost user funds. The only low risk finding reported does not represent any risk.

The following issues were identified during the assessment:

High Risk	Medium Risk	Low Risk
0	0	1

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 strategies recently ported to V3. The changes evaluated consisted in refactoring of code, CRV 3Pool strategies ported from V2, new SwapManager, Uniswap TWAP Oracles integration and bug fixes.

The audit started on May 17 and was conducted on the Git repository at https://github.com/bloqpriv/vesper-pools-v3. The last commit reviewed during this engagement was 93a83985b6fe9be615b081da2138938c4790907f from May 17:

```
commit 93a83985b6fe9be615b081da2138938c4790907f
Merge: 4b96087 ad6817b
Author: patidarmanoj10 <32006767+patidarmanoj10@users.noreply.github.com>
Date:   Mon May 17 12:29:07 2021 -0700

   Merge pull request #121 from bloqpriv/swapman

   Deprecate UniMgr in favor of SwapMgr
```

The scope of the audit was limited the following commits that were added to the project's repository since Coinspect's previous audit and were reviewed during this audit:

```
93a8398 Merge pull request #121 from bloqpriv/swapman
ad6817b Deprecate UniMgr in favor of SwapMgr
4b96087 Merge pull request #118 from bloqpriv/hardhat-test
bbaa5ab added strategy tests for v3
caf5c8a Merge pull request #122 from bloqpriv/deploy
cd2df8e hardthat deploy template
a5893b8 Merge pull request #124 from bloqpriv/time-library
01c9ef2 Created our own version of time library
b31700a Merge pull request #120 from bloqpriv/test-fix
ac8156e Fixed test setup
f73c891 Merge pull request #116 from bloqpriv/minor-updates
16c5bc4 Update multiTransfer, fixed typo and reason string
```

```
e1c3674 Merge pull request #107 from bloqpriv/claimAave
c4469d0 claimRewards, withdrawAllWithEarn
b338f58 Merge pull request #113 from bloqpriv/fixes
33431b6 Fixed logic to maintain withdrawQueue
3e8da42 Merge pull request #105 from bloqpriv/maintainer
65eecaf maintainer
4b4aff2 Merge pull request #106 from bloqpriv/auditfeedback
ecc6041 Audit fixes
e730eaa Merge pull request #80 from bloqpriv/tests
8d3e61d Merge pull request #79 from bloqpriv/minor-fixes
5a43609 Merge pull request #77 from bloqpriv/rename
54bffad Merge pull request #76 from bloqpriv/audit-fix
e35161f Set debtRate, some more test cases, try-catch for withdraw
a4e8d7b Removed redundant modifier, use safeTransfer
826e661 Renamed guardian to keeper
1824828 Merge pull request #72 from bloqpriv/auditFixes
31364db Send value
6689b37 Merge pull request #74 from bloqpriv/patch
2ec820b Removed patch
1a26954 Merge pull request #71 from bloqpriv/hardhat
86592cf Hardhat integration
12debcf Merge pull request #68 from bloqpriv/small-refactor
d1a66e4 Refactored interface imports and code doc
021794c Merge pull request #66 from bloqpriv/usdc-pool
ca79e37 Added USDC pool and aave strategy
```

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 audit focused on their new version of the platform, version 3, for which the framework's most important contracts, those which implement the basic pool and strategy from which all the rest are derived, have been rewritten. The existing pools and strategies have been ported to the new framework.

The most relevant change for this version is the new support for multiple strategy pools. Along with this, the roles in the system have been redesigned.

This incremental audit performed as per Vesper request focused on 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.

The contracts are compiled with Solidity compiler version 0.8.3. Because of a recently published security issue that was fixed in Solidity compiler 0.8.4 it is advised to use this version. See Solidity ABI Decoder Bug For Multi-Dimensional Memory Arrays for more information.

The source code has been extensively documented with commentaries, variables and functions have been renamed, and as a consequence the resulting code is easy to read. Several new tests have been added since the previous audit.

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

3.1 USDC pool and AAVE strategy

Added the USDC pool and related AAVE strategy.

The following commits implement this feature and were reviewed:

- `021794c Merge pull request #66 from bloqpriv/usdc-pool`
- `ca79e37 Added USDC pool and aave strategy`

3.2 Refactoring

These modifications introduced minor code improvements, porting to the new interfaces, removal of now unused imports and contracts, and improved documentation.

The following commits implement this changes and were reviewed:

```
12debcf Merge pull request #68 from bloqpriv/small-refactor
d1a66e4 Refactored interface imports and code doc
01a14eb Updated governed contract
826e661 Renamed guardian to keeper
4b4aff2 Merge pull request #106 from bloqpriv/auditfeedback
ecc6041 Audit fixes
f73c891 Merge pull request #116 from bloqpriv/minor-updates
16c5bc4 Update multiTransfer, fixed typo and reason string
```

3.2 Build and development environment

The following commits are related to the build environment and improved tests:

```
1a26954 Merge pull request #71 from bloqpriv/hardhat
```

```
86592cf Hardhat integration
6689b37 Merge pull request #74 from bloqpriv/patch
2ec820b Removed patch
b31700a Merge pull request #120 from bloqpriv/test-fix
ac8156e Fixed test setup
caf5c8a Merge pull request #122 from bloqpriv/deploy
cd2df8e hardthat deploy template
a5893b8 Merge pull request #124 from bloqpriv/time-library
01c9ef2 Created our own version of time library
4b96087 Merge pull request #118 from bloqpriv/hardhat-test
bbaa5ab added strategy tests for v3
```

3.3 Replaced transfer with sendValue

In order to take into consideration future opcode gas pricing changes, the `sendValue` function is used now to transfer ETH instead of the transfer function. This enables reentrancy attacks as all gas available is passed to the target address. Coinspect auditors evaluated the changes and considered they will not affect Vesper's security.

The following commits implement this feature and were reviewed:

```
1824828 Merge pull request #72 from bloqpriv/auditFixes
31364db Send value
```

3.4 Replaced transferFrom and transfer with safe* version

`PoolShareToken.sol` and `VTokenBase.sol` were improved with this change. The `_whenNotPaused` and `_whenNotShutdown` modifiers were removed from internal functions as they would be called twice.

The following commits implement this feature and were reviewed:

```
a4e8d7b Removed redundant modifier, use safeTransfer
```

3. 5 Added debt rate limit, try/catch around withdraw calls, and new Compound USDC strategy

`VTokenBase.sol` was modified to add the capability to cap the amount a strategy can borrow in a period of time by measuring the number of blocks passed since last rebalance. This is implemented in the `_availableCreditLimit` function:

```
_available = _min(
    (block.number - strategy[_strategy].lastRebalance) *
                                  strategy[_strategy].debtRate,
    _available
);
```

Also, when withdrawing collateral from strategies in the `withdrawQueue`, the withdraw function call has been wrapped in a `try/catch` in order to prevent a revert from stopping the whole withdrawal transaction.

Finally, the new Compound USDC strategy was added.

The following commits implement this feature and were reviewed:

```
54bffad Merge pull request #76 from bloqpriv/audit-fix
e35161f Set debtRate, some more test cases, try-catch for withdraw
```

3. 6 Added new Maintainers role

A new `AddressList` of maintainers and the associated `onlyMaintainers` modifier was added to the `VTokenBase` contract.

The new role has rights to:

1. `updateDebtRatio`
2. `updateWithdrawQueue`

Also, some functions that were accessible by the Governor role are now accessible by the Keepers role instead:

1. `addInList`
2. `removeInList`

Both the Maintainers and the Keepers roles address lists include the Governor that is added to the lists after they are created in the `init` function.

It is worth noting that the Keeper role is able to add or remove Maintainers to the list through the `addInList/removeInList` functions. It is recommended the source code comments for these functions are amended to reflect this fact. These functions are implemented to be generic and allow the Keeper to call the `contains` and `add` functions in any arbitrary address received as a parameter.

Vesper team clarified it is the intended behaviour to let the Keepers add or remove Maintainers.

The following commits implement this feature and were reviewed:

```
3e8da42 Merge pull request #105 from bloqpriv/maintainer
65eecaf maintainer
```

3.7 withdrawQueue improvements

New strategies are now directly added to the `withdrawQueue` when the strategy is created. Also, during migrations, the old one is replaced with the new one instead of adding the new one and keeping the old one in the `strategies` array.
More checks were added to the `updateWithdrawQueue` function to protect from mistakes, however it is still possible for the Maintainer to leave the queue in an inconsistent state as explained in Maintainer could leave withdrawQueue in an inconsistent state.

The following commits implement this feature and were reviewed:

```
b338f58 Merge pull request #113 from bloqpriv/fixes
33431b6 Fixed logic to maintain withdrawQueue
```

3.8 Added new external withdrawAllWithEarn function

A new external function, only callable from the Governor, was added to the Strategy base contract: it claims pending rewards and converts to the collateral token before withdrawing all the funds.
Also, the `_claimRewardsAndConvertTo` function is now called from `_realizeProfit` in `AaveStrategy.sol.`

The following commits implement this feature and were reviewed:

```
e1c3674 Merge pull request #107 from bloqpriv/claimAave
c4469d0 claimRewards, withdrawAllWithEarn
```

3.9 CRV 3Pool strategy ported to V3

The following contracts were ported from Vesper V2 repository: Crv3PoolMgr,

Most changes in these contracts are related to not needing SafeMath in Vesper V3 because of the Solidity compiler version being used. In Crv3PoolMgr the `_depositDaiToCrvPool` function was removed. Crv3PoolStrategy was heavily simplified by using the V3's Strategy interface.

While reviewing these commits, Coinspect became aware of an issue recently fixed in the yearn.finance project's Curve strategy, documented in https://github.com/yearn/yearn-security/blob/master/disclosures/2021-05-14.md. After a quick review, it was determined that the vulnerability described could also affect Vesper as the Crv3PoolStrategy `totalValue` function calls getLpValueAs which relies on Curve's `THREEPOOL.calc_withdraw_one_coin`. **Coinspect contacted the Vesper to warn them about this. The Vesper team was already aware of the incident disclosure and was discussing it internally.**

The following commits implement this feature and were reviewed:

```
9d1fd69 Merge pull request #104 from bloqpriv/portcrv
10fdafb Port 3Pool Strategy
5355f25 update package.json
```

3.10 All swaps performed through new upgradable SwapManager

The `UniswapManager.sol` contract was removed from the repository. It has been replaced with the new `SwapManager` and all token swaps (or expected swap value) are now performed through this contract.

The Strategy base contract has been modified, now its constructor gets passed the address of a SwapManager contract, which can be updated afterwards by the Governor role with the new `updateSwapManager` function. All derived strategies have been modified accordingly. This will allow upgrading the SwapManager when needed.

Also, the `_safeSwap` function now has a new `_minAmountOut` parameter, which is set to 1 if the value passed by the caller is 0.

In order for the new SwapManager to function, strategies now have to grant token allowances for all the exchanges configured in the `SwapManager`. The `safeApprove` function invocations have been replaced with `approve` ones.

The SwapManager contract itself is the same that was reviewed in Coinspect's last audit of the Vesper V2 platform. The `SwapManager` contract allows comparing the expected trade result between a set of configured exchanges. The currently configured AMMs are `Uniswap` and `Sushiswap`. In this version a new feature was added: a **TWAP oracle factory**.

A full analysis of Time-Weighted Average Price oracles and their security properties are outside of scope for this engagement. Detailed information is provided in the following links:

1. https://uniswap.org/docs/v2/core-concepts/oracles
2. https://uniswap.org/audit.html#org794ac51

A mapping of oracles (indexed by token pair and time period) is maintained in this contract. Anybody is able to create a new oracle by providing: tokenA, tokenB, period of time and DEX index. The new oracle implementation (OracleSimple) is instantiated using the token pair obtained from the desired DEX from the list of configured exchanges in SwapManager.

Anybody can consult the oracles in the SwapManager. There are to available options:

1. Consult for free view function
2. Consult and pay for the oracle update

The `consult` function retrieves the amount of tokens that would be obtained in an exchange and then updates the oracle. **Depending on how this functionality is being used, it might be better to update the oracle first and then consult it.**

`OracleSimple` is based on Uniswap's sample implementation:

uniswap-v2-periphery/ExampleOracleSimple.sol

The code was modified to enable the creation of oracles with different time periods. Also, a new state variable named `isStale` was introduced that is not being used so far.

The TWAP Oracle functionality is not yet used by Vesper's contracts. Coinspect recommends their security is fully evaluated once they are integrated.

The following commits implement this feature and were reviewed:

```
93a8398 Merge pull request #121 from bloqpriv/swapman
ad6817b Deprecate UniMgr in favor of SwapMgr
```

3.11 External contracts addresses

The following addresses referencing external contracts are used and were verified to be correct:

1. `0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48` (USDC as listed in https://www.centre.io/developer-resources)

2. `0xBcca60bB61934080951369a648Fb03DF4F96263C` (`aUSDC Token V2` as listed in https://docs.aave.com/developers/deployed-contracts/deployed-contracts)
3. `0x39AA39c021dfbaE8faC545936693aC917d5E7563` (cUSD as listed in https://compound.finance/docs)
4. `0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D` (`UniswapV2Router02` as listed in https://uniswap.org/docs/v2/smart-contracts/router02/)
5. `0xd9e1cE17f2641f24aE83637ab66a2cca9C378B9F` (`SushiV2Router02` as listed in https://dev.sushi.com/sushiswap/contracts)
6. `0x5C69bEe701ef814a2B6a3EDD4B1652CB9cc5aA6f` (`UniswapV2Factory` as listed in https://uniswap.org/docs/v2/smart-contracts/factory/)
7. `0xC0AEe478e3658e2610c5F7A4A2E1777cE9e4f2Ac` (`SushiV2Factory` as listed in https://dev.sushi.com/sushiswap/contracts)

4. Summary of Findings

ID	Description	Risk	Fixed
VSP-035	Maintainer could leave withdrawQueue in an inconsistent state	Low	✘

5. Findings

VSP-035	Maintainer could leave withdrawQueue in an inconsistent state	
Total Risk Low Fixed ✘	Impact Low Likelihood Low	Location VTokenBase.sol

Description

A Maintainer could leave the `updateWithdrawQueue` in an inconsistent state by mistake.

Several checks were added to the `updateWithdrawQueue` function, however it is still possible to update the queue so it has the appropriate length, but all the strategies are the same:

```
function updateWithdrawQueue(address[] memory _withdrawQueue) external onlyMaintainer {
    uint256 _length = _withdrawQueue.length;
    require(_length > 0, "withdrawal-queue-blank");
    require(_length == withdrawQueue.length && _length == strategies.length,
            "incorrect-withdraw-queue-length");
    for (uint256 i = 0; i < _length; i++) {
        require(strategy[_withdrawQueue[i]].active, "invalid-strategy");
    }
    withdrawQueue = _withdrawQueue;
}
```

This might have unexpected side effects in other parts of the code that expect the strategies in the queue to be unique. For example, the `migrateStrategy` function only replaces the first occurrence of the old strategy in the `withdrawQueue`:

```
function migrateStrategy(address _old, address _new) external onlyGovernor {

     ...
```

```
        // Strategies and withdrawQueue has same length but we still want
        // to iterate over them in different loop.
        for (uint256 i = 0; i < strategies.length; i++) {
            if (strategies[i] == _old) {
                strategies[i] = _new;
                break;
            }
        }
        for (uint256 i = 0; i < withdrawQueue.length; i++) {
            if (withdrawQueue[i] == _old) {
                withdrawQueue[i] = _new;
                break;
            }
        }
    }
```

This would leave a strategy set to non-active in the withdrawal queue.

Coinspect did not find any exploitable scenario derived from this issue in the current code.

Recommendation

Consider preventing duplicates in the `withdrawQueue` or document that this is possible.

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.