Vesper Pools Security Audit

coinspect

Smart Contract Audit
Vesper Pools

Prepared for Bloq • January 2021

Fourth Review v210119

1. Executive Summary

In January 2021, Bloq engaged Coinspect to perform a fourth source code review of their new 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 **Jan 6th**.

The following issues were identified during the assessment:

High Risk	Medium Risk	Low Risk
0	**2**	**3**

Two medium risk vulnerabilities were found during this audit. The first one, vVSP governance token lock period extended after new mint, is related to how Vesper keeps track of locked governance tokens and the second one, Attacker can prevent non-atomic strategies upgrade, exploits a weakness in the strategy upgrade mechanism. Also, three low risk findings are reported: VSP strategy broken rebalance friction check details a mistake in the rebalance friction verification and the other two ones are minor code quality issues.

In January 2021, Coinspect updated the fix status for each finding with Vesper's feedback.

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 AAVE V2 Maker strategy. The changes evaluated consisted in refactoring of the governance tokens, gas optimizations and bug fixes. The new strategy resembles the existing AAVE Maker strategy, with minor changes to use AAAVE v2.

The audit started on January 11th and was conducted on the Git repository at https://github.com/bloqpriv/bpools. The last commit reviewed during this engagement was `651258dced7d5578683274c993246e199495d0d5` from **January 6th**:

```
commit 651258dced7d5578683274c993246e199495d0d5
Merge: 91720b4 e2f5969
Author: Rohit Solia <rohit.solia@gmail.com>
Date:   Wed Jan 6 10:47:31 2021 -0600

   Merge pull request #362 from bloqpriv/withdraw-lock

   vVSP withdraw lock mechanism
```

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

```
c3693a6bd7d6a657ae716c6d8d5bf9c32bb47db417102b19a670dceaf23591af  contracts/VSP.sol
6a45ece01a41666b267f37688525199d2418517f017a3158a2ed9490a7dbea45  contracts/governor/GovernanceToken.sol
d6cadeb94011b7a7662dec42794dbe9afc31655510d75459807b020f8d302f2f  contracts/governor/GovernorAlpha.sol
a4c0a5611d3255c6e56b1e9c6f1c5b0ffd3aa90a1b6ecd2fec36f0853c78ae48  contracts/governor/VVSP.sol
e1ab9f60e2b8eadf1021d2f839b6e0e3d3e789a4eb33748ce19cdf2f15cd6496  contracts/strategies/AaveMakerStrategy.sol
81bf3c117bc61ba13bbd171b0fe2adaff1e818856e60b2c531f4a133dc1976e7  contracts/strategies/AaveV2MakerStrategy.sol
3c671cb4d8c9e1133c7fe23e66df56d97a2a97bdad86e36b992acc4efa1c0b74  contracts/strategies/CompoundStrategy.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 Yearn. A full review comprising these interactions should be performed in order to fully assess the security of the project.

3. Assessment

Vesper's end goal is to become an AI-driven resource management engine for the tokenized world. The code reviewed during this engagement implements crypto asset pools that enable users to generate earnings by supplying 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. These deposits are in turn deposited by the pool in MakerDAO to generate the DAI stable coin.

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

3.1 Governance token changes

Commit `66b96260ae9bc0432cf07cfa8bdc11171671696e` introduced several modifications. This commit moved the governance functionality previously in the VSP token to the vVSP token and includes several additional modifications to the code.

The following files were removed:

1. `pools/VVSP.sol`
2. `token/GovernorAlpha.sol`
3. `token/VSP.sol`
4. `token/VSPRGovernanceToken.sol`

The governance related code in the Vesper Token was removed from the new `VSP.sol` (former `VSPR.sol`). In concrete, all calls to `_moveDelegates` were eliminated from this file.

The new `governor/GovernanceTokens.sol` contract is based on Compound's COMP.sol. Its main goal is to allow token holders to delegate voting rights. It replaces the previous `VSPRGovernanceToken.sol` which was an ERC20 token, now it is a `PoolShareToken` instead. The vote delegation functionality is implemented as a checkpointing mechanism that keeps track of how many votes each delegate has at a determined moment in time. Even though there is no maximum amount of checkpoints that can be created, and there exists the possibility of running out of gas during checkpoint processing, Coinspect did not find any exploitable scenario in the current context.

The new `governor/VVSP.sol` source file was added. The `VVSP` token is a `GovernanceToken`, which in turn is a `PoolShareToken`. The calls to `_moveDelegates` that were removed from `VSP.sol` were added here in the `_beforeTokenTransfer` ERC20 hook.

This allows to account for VVSP delegates when moving tokens through `mint`, `burn`, `transfer` and `transferFrom` operations take place.
It is worth noting that because the `_beforeTokenTransfer` implementation in `VVSP.sol` does not invoke the parents implementation of the hook, the rewards are not updated in `PoolShareToken.sol`. As a result, vVSP tokens do not pay rewards.

3.2 Code quality and gas optimization changes

The following commits were reviewed:

1. Commit `6c9ade83b703613f50758d0316d249269813a221` changed several functions visibility to external in `GovernorAlpha.sol`.
2. Commit `75ca166007812728b57eec985dffe997eb51f0dc` optimized code gas usage in the Compound strategy implementation.

Coinspect verified the modifications introduced in these commits are correct.

3.3 Pool logic fix

Commit `07badfc73bca19bf8e8d235ca390564739e80ca4` fixed a bug in the pool resurface logic. This bug was discovered by Vesper's team and consisted in more collateral than needed being transferred from the pool to the strategy during a resurface scenario. As a consequence, part of this collateral would remain in the strategy after the tokens being swapped via Uniswap.

Coinspect verified the updated code correctly addresses this issue.

3.5 New AAVE V2 Maker strategy

The new file `contracts/strategies/AaveV2MakerStrategy.sol` was added to the project. Coinspect reviewed the few differences with the existing `AaveMakerStrategy.sol` file, which were all minor changes needed to support the AAVE V2 interface and do not affect the security of the previously reviewed strategy.

4.6 vVSP governance tokens lock mechanism

Commit `e2f5969981bc29c810a17aa5f48798066d3b0f3f` added a lock mechanism for vVSP tokens. The tokens are only allowed to be transferred or withdrawn after a lock period has passed since the moment they were minted.
Because of the way the lock mechanism works, a user might end up with his tokens locked more time than expected, as described in vVSP governance token lock period extended after new mint.

4. Summary of Findings

ID	Description	Risk	Fixed
VSP-016	vVSP governance token lock period extended after new mint	Medium	✘
VSP-017	Unused variables in AaveV2Strategy	Low	✔
VSP-018	Strategy's name and version should be declared constant	Low	✔
VSP-019	VSP strategy broken rebalance friction check	Low	✔
VSP-020	Attacker can prevent non-atomic strategies upgrade	Medium	✘

5. Remediations

During January 2021 Coinspect verified the findings that Vesper decided to address had been correctly fixed.

The following table lists the findings that were fixed with their corresponding fix status and commit:

ID	Description	Status
VSP-016	vVSP governance token lock period extended after new mint	Won't fix
VSP-017	Unused variables in AaveV2Strategy	Fixed `60efdd908c0caa8d30939be0e85d151d70c24ded`
VSP-018	Strategy's name and version should be declared constant	Fixed `fbf0be0acc084c89bfd93d3755ec66544363988e`
VSP-019	VSP strategy broken rebalance friction check	Fixed `65769ed69b0f94b8e2e39fb9f74f245a537dd20e`
VSP-020	Attacker can prevent non-atomic strategies upgrade	Will fix

More detailed status information can be found in each finding.

6. Findings

VSP-016	vVSP governance token lock period extended after new mint

Total Risk	Impact	Location
Medium	Medium	VVSP.sol
Fixed	**Likelihood**	
✘	High	

Description

The `vVSP` governance tokens can only be withdrawn or transferred after a lock period has passed since they are minted. The lock period is set to 1 day initially in the constructor, but the contract controller can change it anytime with the `updateLockPeriod` function.

The `vVSP` governance tokens are implemented as `PoolShareTokens` with an overridden `_beforeTokenTransfer` function implementation:

```
function _beforeTokenTransfer(
    address from,
    address to,
    uint256 amount
) internal override {
    if (from == address(0)) {
        // Token being minted i.e. user is depositing VSP
        // NOTE: here 'to' is same as 'msg.sender'
        depositTimestamp[to] = block.timestamp;
    } else {
        // transfer, transferFrom or withdraw is called.
        require(
            block.timestamp >= depositTimestamp[from].add(lockPeriod),
            "Operation not allowed due to lock period"
        );
```

When tokens are minted, the block timestamp is stored for the address that minted them. When tokens want to be transferred or withdrawn, this timestamp plus the configured lock period is compared to the current block's one in order to decide if the operation is going to be allowed or denied.

If the user mints more tokens before the lock time is due, and then needs to withdraw the tokens that were minted the first time, the operation will be denied because only the last mint timestamp is stored. This results in tokens being locked more time than intended and could have negative consequences for the minter if the price of the token is decreasing.

Recommendations

Keep record of the amount of tokens and the corresponding timestamp each time tokens are minted.

Status

Vesper's team decided not to fix this issue as right now only the contract owner would be affected.

VSP-017 Unused variables in AaveV2Strategy

Total Risk	Impact	Location
Low	Low	AaveV2Strategy.sol
Fixed	Likelihood	
✔	Low	

Description

The following two variables are declared in AaveV2Strategy but never used:

```
address internal constant ETH = 0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE;
address internal constant WETH = 0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2;
```

Recommendation

Remove the unnecessary variable declarations.

Status

This issue was fixed in commit `60efdd908c0caa8d30939be0e85d151d70c24ded`.

VSP-018	Strategy's name and version should be declared constant

Total Risk	Impact	Location
Low	Low	strategies/*Strategy*.sol
Fixed	**Likelihood**	
✔	Low	

Description

The state variables name and version in every strategy contract are not declared as constant. For example, in `VSPStrategy.sol`:

```
string public name = "Strategy-VSP";
string public version = "1.0.3";
```

Declaring these variables as constant reduces gas cost for each contract.

Recommendation

Declare the above mentioned state variables as constant.

Status

This issue was fixed in commit `fbf0be0acc084c89bfd93d3755ec66544363988e`.

VSP-019 VSP Strategy broken rebalance friction check

Total Risk	Impact	Location
Low	Low	VSPStrategy.sol
Fixed	**Likelihood**	
✔	High	

Description

The Vesper Treasury is paid fees from all Vesper pools. These fees are paid in interest-earning pool shares which get sent to the vVSP pool.

The VSP strategy `rebalance` function's goal is to trigger earnings rebalancing from each of the available pools (but the vVSP pool itself). It can only be called by the vVSP pool, where it can be triggered by anyone by calling the vVSP pool `rebalance` function.

Each of the pool's earnings (in the form of pool's shares) are liquidated and then are swapped for more VSP tokens in Uniswap. From these VSP tokens, the founders fee is paid to the founders vault and the rest is deposited into the vVSP pool, in this way distributing them to all the vVSP pool participants.

The function has a check to prevent it from being called too often, where a configurable rebalance friction parameter, configured in the controller contract, is enforced:

```
function rebalance() external {
    require(msg.sender == address(vvsp), "Caller is not vvsp pool");
    require(
            block.number - lastRebalanceBlock >= controller.rebalanceFriction(address(vvsp)),
        "Can not rebalance"
    );
```

Because the `lastRebalanceBlock` variable is never initialized and never updated, the condition in the require statement will always result true and all calls to `rebalance` will be executed ignoring the rebalance friction validation.

Recommendation

Update the `lastRebalanceBlock` to the current block number each time the rebalance is performed by adding the following line to the function:

```
lastRebalanceBlock = block.number;
```

Status

This issue was fixed in commit `65769ed69b0f94b8e2e39fb9f74f245a537dd20e`.

VSP-020 Attacker can prevent non-atomic strategies upgrade

Total Risk	Impact	Location
Medium	Medium	VSPStrategy.sol strategies/*Strategy*.sol
Fixed	Likelihood	
✘	High	

Description

Vesper strategies can be upgraded to new ones, this can be performed by the controller contract:

```
function updateStrategy(address _pool, address _newStrategy)
    external
    onlyOwner
    validPool(_pool)
{

     ...

    if (currentStrategy != address(0)) {
        require(IStrategy(currentStrategy).isUpgradable(), "strategy-is-not-upgradable");
        vpool.resetApproval();
    }
     strategy[_pool] = _newStrategy;
```

The `updateStrategy` function verifies the strategy being replaced is upgradable by calling the strategy's `isUpgradable` function. Most strategies consider themselves upgradable when the total collateral token locked in the pool is zero. For example, in the `VSPStrategy` the function is implemented this way:

```
function isUpgradable() external view returns (bool) {
    return IERC20(vvsp.token()).balanceOf(address(this)) == 0;
}
```

Then, the strategy update flow consists in 2 separate actions:

1. First, funds are withdrawn from the strategy being replaced, usually done by calling the strategy's `withdrawAll` function, usually though the controller contract's `executeTransactions` function;
2. Then the controller `updateStrategy` function is called in order to change the strategy to a new one

Because these 2 actions are not atomic, if an attacker transaction transferring tokens (any amount will suffice) to the strategy is inserted before the update takes place, the update will revert. An attacker can continue doing this to prevent the strategy from being updated.

Also, because of the way `executeTransactions` is implemented, it is not possible to call the controller's `updateStrategy` function from the controller itself right after calling `withdrawAll`.

Even though this issue was investigated while reviewing the VSP strategy, it affects the other strategies in the project as well.

Recommendations

Implement an atomic strategy update mechanism in order to prevent transactions between the moment the tokens are withdrawn and the strategy is switched over.

Status

The Vesper team has stated they will be fixing this issue as soon as possible.

7. Conclusion

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

However, during this audit Coinspect observed the new governance token lock period mechanism could result in tokens being locked more time than intended in certain scenarios and that the strategy update mechanism could be prevented by an attacker by front-running the update transactions sent by the controller.

8. Disclaimer

The information presented in this document is provided as is and without warranty. Vulnerability assessments are a “point in time” analysis and as such it is possible that something in the environment could have changed since the tests reflected in this report were run. This report should not be considered a perfect representation of the risks threatening the analysed system, networks, and applications.