Vesper Earn
Smart Contract Audit

coinspect

Vesper Earn

Smart Contract Audit

v211112

Prepared for Vesper • September 2021

1. Executive Summary

In September 2021, Vesper engaged Coinspect to perform a source code review of Vesper's v2 and v3 pools and strategies. The objective of the project was to continue to evaluate the security of the smart contracts.

The changes evaluated included several integrations with protocols from the Ethereum DeFi ecosystem such as: Alpha Homora, Rari Fuse, Yearn, Compound, AAVE and Convex Finance protocols and a new class of strategies called "Vesper Earn" besides other code changes.

Overall, Coinspect did not find any high risk security vulnerability introduced by the modifications made to Vesper since its previous audit that would result in stolen or lost user funds. However, three medium risk (high impact, but low likelihood) and one low risk issue are reported . Some of these issues are caused by the lack of proper error handling or excessive trust when interacting with external contracts. One of the issues reported calls attention to the lax access controls being applied to the reward tokens distribution parameters.

The following issues were identified during the assessment:

High Risk	Medium Risk	Low Risk
0	**3**	**1**
Fixed 0	Fixed 2	Fixed 1

2. Assessment and Scope

This incremental audit continued the previous efforts evaluating Vesper's smart contracts and focused on the following contracts and features as per Vesper's team request:

1. **Vesper v2**
    a. Modifications introduced in order to protect from slippage
2. **Vesper v3**
    a. New PoolRewards contract
    b. New CompoundLeverage* strategies
    c. New CompoundXY* strategies
    d. New Convex strategies
    e. Modifications to Vesper Earn strategy framework
    f. New Vesper Earn strategies
        i. EarnCompound*
        ii. EarnCream*
        iii. Earn*Maker*
        iv. EarnAave*
        v. EarnYearn*
        vi. EarnRariFuse*
        vii. EarnAlphaLend*
3. **VFR - Vesper Fixed-Rate Retargeting**

The audit started on September 20 and was conducted on the master branch of the git repositories at:

1. https://github.com/bloqpriv/vesper-pool as of commit `eda8534011f427a7c7aff025664530087a7e536a` of September 9, 2021.
2. https://github.com/bloqpriv/vesper-pools-v3 as of commit `39332ea67e76040e43a0c2ee243d962ca656ad64` of September 21, 2021.

The scope of the audit was limited to the following Solidity source files in the Vesper v3 repository, shown here with their `sha256sum` hash:

```
d377bbe3bc7b9dcdfa3b3d238d6f69989c68612e1a51908e66b2803c439367b5  ./pool/earn/EarnDrip.sol
7a6b870b37848a5a7ff4dc002a58aa5fe01b6d5fc2105b45f978e17bc662b601  ./strategies/yearn/earn/EarnYearnStrategyETH.sol
9a5066b2956d8239f830d8a9b62da809f9352dfcc6a3cb312a07edaa7e6d3693  ./strategies/yearn/earn/EarnYearnStrategy.sol
c9e2ffd0adeaa99ac6c93f6c40d3ae07493c3a23fb83d761f330dac9bc4720d2  ./strategies/Earn.sol
50f89f77e7424be6b192f99cb6f2da6592c2edb23a53f8b53975c35cb980860c  ./strategies/maker/EarnVesperMakerStrategyETH.sol
67a05851cfaf877c4d7ebc19eef30bc2e5c4a61bc9f422b5810e873f1c64bbc9  ./strategies/maker/EarnCompoundMakerStrategyETH.sol
```

```
5bd3c58cc0e187ba94265b4b126ec695046a541f9bf460e30137bc42d00f54d4  ./strategies/maker/EarnCompoundMakerStrategy.sol
48208533f1fd9548cfd6dc27402d39aa5e06e29611f9ec8ed28686f33bb4c5a4  ./strategies/maker/EarnAaveMakerStrategyETH.sol
39adfeb92e600060866f59b8aef331e408bb27036d6fae4613b67cda2d4562d3  ./strategies/maker/EarnVesperMakerStrategy.sol
5f7fcad9e668e8dcb0fa459cec3e01694172fb0907cbcb4f261cb9d38c620930  ./strategies/maker/EarnAaveMakerStrategy.sol
9c6c67ae387c9e6d8e57b036500976ea004c38852b7a1aa70963ca7a46b7b253
./strategies/compound/earn/EarnCompoundStrategyETH.sol
c5f461dd67431e8c7bcff91fa2314c011662234bb93ff48956f2b26ca5552ea2  ./strategies/compound/earn/EarnCompoundStrategy.sol
31e4c1d0507b56777432e4b2355f1e7fd42259c92faf155be7066f4b0bbb1a8b  ./strategies/compound/CompoundXYStrategyETH.sol
491fdbb1fc8a10d05660644057ada3e0169e988fc14822b917b3c5b34bb4e0a9  ./strategies/compound/CompoundLeverageStrategy.sol
8a528e842be22f537f1e60ee31e09a474e7cbfaf254a4b825d6a12b2473cafba  ./strategies/compound/CompoundXYStrategy.sol
f031b9522319c64667f6e8f921e6d198074b8554830ace0f2c04dab53e81053c
./strategies/compound/CompoundLeverageStrategyETH.sol
06eb254c8da31154014910245ffe1350b37d11172deb038d0588ceb147176d88  ./strategies/cream/earn/EarnCreamStrategyETH.sol
fd010a122732ef2676d2c84c3f0ab386620ab28618c1f55fe309826f053d8c5a  ./strategies/cream/earn/EarnCreamStrategy.sol
ecd77a1148155e8e0a950224ad0ee92a068ada59f71f0742090fb8c31fab2eff  ./strategies/convex/ConvexSBTCStrategyWBTC.sol
d036b0176eaec8f47da345364a2abfaae120bf50db6f7bf9355079a7fb48365e  ./strategies/convex/ConvexStrategyDAI.sol
278a1a3b349767b240fb0b617f4faee2600c6bd74cd13fc5cbb02965af906e3d  ./strategies/convex/ConvexStrategy.sol
c549b689c8b3ac84bea8bb5538ebb6d62530e5738bb7eec978a7ef99fafc9dc0  ./strategies/aave/earn/EarnAaveStrategy.sol
36399c3ce89e9042f5f82c1b7e69a039e4bb425836ae9c36e79140c0e9503d54  ./strategies/aave/earn/EarnAaveStrategyWETH.sol
d4ea31f2b915441fe84c30b8e2feb44d6aabbdcd6ae14f36332114ad9dbe9237
./strategies/rari-fuse/earn/EarnRariFuseStrategyETH.sol
e422623a13ae14ee3d892313d9a0e43728c84e7e8380431623a74503d36c39a7
./strategies/rari-fuse/earn/EarnRariFuseStrategy.sol
9c3816581b9dd352117a6a8b5629ffe317a7e5c6b084fc11825a992b7ace01e8
./strategies/alpha/earn/EarnAlphaLendStrategyETH.sol
ce67414e71aaf2b009641642f7f63ac47b2faaa544112cc4b635f9ade3d49388  ./strategies/alpha/earn/EarnAlphaLendStrategy.sol
717ba81d0e465ce0b8ef8fe69fd90261e8492ec43da3284ebdf0b1388bc09985  ./strategies/VFR.sol
1617640211492845b43cb6bd373702301229e869a14ec5a8566dd8ebe8bcadf1  ./pool/vfr/VFRBuffer.sol
0dc2d1b4e5d701e67034908ba772f9023480c294ed6be3ba5a5b763971019d69  ./pool/vfr/VFRCoveragePool.sol
29b16cf72160177f560785a061a3fce9b9140a84fcb0eb0a69a82bd3fc50382d  ./pool/vfr/VFRPool.sol
e032e8cc73cb576c0e5f10b4a98b3dd677c5dfec607a77ab0ea8184e4f0f6ee2  ./pool/vfr/VFRStablePool.sol
81afc05243ac2f85e22572b26a41d51ec4fa78dbb42548928d57a9194516c23f  ./strategies/compound/vfr/CompoundCoverageStrategy.sol
333b728683677a26b9be0fca318d91d8d19a477bb633a626b3ddd2f87c37e26a  ./strategies/compound/vfr/CompoundCoverageStrategyDAI.sol
4d3d29d0296185c67271377b002da8dd3d4dbdb33a40d9f4eae268e17b0badbe  ./strategies/compound/vfr/CompoundStableStrategy.sol
03560f4cc1d078513d9187db3f2556778b90e33eb867a9fb2b1a80d834a7f1ed  ./strategies/compound/vfr/CompoundStableStrategyDAI.sol
8b3042bb70a7ca9869363c24044efb1493eda03c6c238c86084186cf44c7de42  ./strategies/convex/vfr/ConvexCoverageStrategy.sol
6943d2cfda9609c1204ae6638ee1dc76c3915c23e404690cc9760e17a44486a7  ./strategies/convex/vfr/ConvexCoverageStrategyDAI.sol
c043721d4b02eeeb3766f58b65d48807db7abc8abf6c497b0d237b26d15b1b5a  ./strategies/convex/vfr/ConvexStableStrategy.sol
0ffa088bb3d340d8f8b237fdb063b8311c298e8aea12ac37a142c542d6af8a0c  ./strategies/convex/vfr/ConvexStableStrategyDAI.sol
```

These are the files audited in Vesper v2's repository with the corresponding `sha256sum` hash:

```
a01b6e8b3dccb8285b1efe9fbe2128951d912c8be293c06c8fd2c3d77765e8c8  strategies/OraclesBase.sol
88d1e49dae3352ac4aa4535202c5d98133b721612bbaa5795889d318d594bcb1  strategies/VSPStrategy.sol
```

On November 11, updated code was reviewed by the Coinspect team. The current status of each finding is further detailed below each issue. The issue VSP-38 was dismissed after further analysis.

These smart contracts interact with multiple third-party projects which were not part of this review, including but not limited to: MakerDAO, Uniswap, Compound, AAVE, and Convex. A full review comprising these interactions should be performed in order to fully assess the security of the project.

The following sections detail the modifications and new features reviewed.

Vesper v2

The contract `OraclesBase` was added. This contract introduces oracle based slippage protection. Though, by default it is disabled, as the allowed slippaged is initialized to 100%. Only the controller role is allowed to update the oracle's configuration: slippage, period and data source. The contract is able to detect stale oracles and returns an error value in that scenario.

The VSPStrategy contract was modified to utilize the new OraclesBase contract in the `_rebalanceEarned` function to calculate the minimum expected amount of tokens accepted when swapping taking into account the configured slippage.

Vesper v3

Regarding the PoolRewards contract:

1. The Pool governor role can create new rewards.
2. Anybody can claim rewards.
3. The functions in this contract are protected from reentrancy.
4. Rewards are based on the Pool's total supply and user's account balance.
5. Only the Pool can update rewards for an account. The function `_updateReward` is called from claimRewards too.
6. Coinspect verified that upon pool share token transfers, the accumulated rewards can not be claimed by both the account transferring and the account receiving the tokens. The pool share token transfer calls updateReward itself for both sender and destination account, which in turns updates userRewardPerTokenPaid.

It is worth noting that for users to claim rewards, enough balance has to be available in the PoolRewards contract. If the contract cannot fulfil the user's claim request no error message is provided to the users, and it is suggested to provide it.

With respect to the newly introduced strategies (CompoundLeverage, CompoundXY, and Convex*) Coinspect found a weakness shared by all of them: when calling external contracts, the return values are not checked by the caller and this could lead to different unexpected scenarios as described in Functions silently

failing when interacting with Compound and Functions silently failing when interacting with Convex.

It was also found that the CompoundXY strategy sets an infinite approval to the Compound contract project when the borrow token is updated as described in External protocols security incident could result in funds drained from strategies.

The Earn (derived from the Strategy base contract) and EarnDrip (derived from PoolRewards) contracts allow the creation of a new set of strategies that forward their earnings by depositing them to grow pools and then depositing the shares obtained into a drip contract. An issue with lax access controls in EarnDrip was found and detailed in Lax access controls to reward token distribution configuration. Several Earn derived strategies were reviewed: the original strategy's realizeProfit function was modified in all of them. This function now converts gains from the collateral to the drip token and calls the forwardEarning function provided by the Earn contract.

VFR - Vesper Fixed-Rate Retargeting

`VFR` is a system where two pools interact through a buffer: a stable pool and a coverage pool.

The stable pool sets an `APY` and tries to accomplish it. It does so by requesting funds to the buffer when it is below the target and sending the extra profits when it is above it.

The coverage pool fills the buffer using the profits until a threshold based on the APY and a coverage time is reached. If the buffer is over the threshold the extra funds are requested and considered profits by the coverage pool.

Coinspect noted that if the ERC20 tokens used have a fee on transfer, it might break the stable pool when losses are below the expected APY (see `VSP-42`).

External contracts addresses

The following addresses referencing external contracts are used in the contracts reviewed were verified to be correct:

1. 0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2 for WETH9 as shown in https://etherscan.io/address/0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2#code
2. 0xc00e94cb662c3520282e6f5717214004a7f26888 for COMP as listed in https://compound.finance/docs
3. 0x3d9819210A31b4961b30EF54bE2aeD79B9c9Cd3B for Comptroller as listed in https://compound.finance/docs
4. 0x4Ddc2D193948926D02f9B1fE9e1daa0718270ED5 for cETH as listed in https://compound.finance/docs
5. 0x6B175474E89094C44Da98b954EedeAC495271d0F for DAI as listed in https://github.com/makerdao/developerguides/blob/master/dai/dai-token/dai-token.md
6. 0xbEbc44782C7dB0a1A60Cb6fe97d0b483032FF1C7 for 3Pool as listed in https://curve.readthedocs.io/ref-addresses.html
7. 0x6c3F90f043a72FA612cbac8115EE7e52BDe6E490 for 3Pool as listed in https://curve.readthedocs.io/ref-addresses.html
8. 0xbFcF63294aD7105dEa65aA58F8AE5BE2D9d0952A for 3Pool Gauge as listed in https://curve.readthedocs.io/ref-addresses.html
9. 0x7fC77b5c7614E1533320Ea6DDc2Eb61fa00A9714 for sBTC pool as listed in https://curve.readthedocs.io/ref-addresses.html
10. 0x075b1bb99792c9E1041bA13afEf80C91a1e70fB3 for sBTC pool as listed in https://curve.readthedocs.io/ref-addresses.html
11. 0x705350c4BcD35c9441419DdD5d2f097d7a55410F for SBTC gauge as listed in https://curve.readthedocs.io/ref-addresses.html
12. 0x1b40183EFB4Dd766f11bDa7A7c3AD8982e998421 for VSP token as listed in https://docs.vesper.finance/vesper-grow-pools/vesper-grow/audits
13. 0xe382d9f2394A359B01006faa8A1864b8a60d2710 for SwapManager deployed by Vesper as shown in https://etherscan.io/address/0xe382d9f2394A359B01006faa8A1864b8a60d2710
14. 0xbFcF63294aD7105dEa65aA58F8AE5BE2D9d0952A for 3Pool liquidity gauge as listed in https://curve.readthedocs.io/ref-addresses.html
15. 0x4e3FBD56CD56c3e72c1403e103b45Db9da5B9D2B for CVX as listed in https://docs.convexfinance.com/convexfinance/faq/contract-addresses

16. 0xF403C135812408BFbE8713b5A23a04b3D48AAE31 for Booster as listed in https://docs.convexfinance.com/convexfinance/faq/contract-addresses
17. 0xccF4429DB6322D5C611ee964527D42E5d685DD6 for cWBTC2 as listed in https://compound.finance/docs
18. 0x5d3a536E4D6DbD6114cc1Ead35777bAB948E3643 for cDAI as listed in https://compound.finance/docs
19. 0x030bA81f1c18d280636F32af80b9AAd02Cf0854e for aWETH as listed in https://docs.aave.com/developers/deployed-contracts/deployed-contracts
20. 0x028171bCA77440897B824Ca71D1c56caC55b68A for aDAI as listed in https://docs.aave.com/developers/deployed-contracts/deployed-contracts
21. 0xD06527D5e56A3495252A528C4987003b712860eE for crETH as listed in https://github.com/CreamFi/cream-docs/blob/ethereum-mainnet/lending/lending-contract-address.md
22. 0xa258C4606Ca8206D8aA700cE2143D7db854D168c for yvBOOST-ETH as listed in https://yearn.finance/vaults/0xa258C4606Ca8206D8aA700cE2143D7db854D168c
23. 0xeEa3311250FE4c3268F8E684f7C87A82fF183Ec1 for WETH SafeBox as listed in https://alphafinancelab.gitbook.io/alpha-homora-developer-doc/become-the-leader-of-alpha-homora-v2

3. Summary of Findings

Id	Title	Total Risk	Fixed
VSP-39	Functions silently failing when interacting with Convex	Medium	✔
VSP-40	External protocols security incident could result in funds drained from strategies	Low	✔
VSP-41	Lax access controls to reward token distribution configuration	Medium	✘
VSP-42	Broken accounting on special ERC20	Medium	✔

4. Detailed Findings

VSP-39 Functions silently failing when interacting with Convex

Total Risk	Impact	Location
Medium	High	`ConvexStrategy.sol`
Fixed	Likelihood	
✔	Low	

Description

The `ConvexStrategy` contract fails to check the return values from external calls made to Compound contracts and this could lead to unexpected scenarios not being properly handled and the incorrect accounting of funds in the strategies involved.

As detailed in the Convex's documentation (located in Booster - ConvexFinanceIntegration and BaseRewardPool - ConvexFinanceIntegration), both `Booster` and `BaseRewardPool` interface functions return a boolean value.

For example, in the `ConvexStrategy` contract:

```
function _stakeAllLp() internal override {
    uint256 balance = IERC20(crvLp).balanceOf(address(this));
    if (balance != 0) {
        IConvex(BOOSTER).deposit(convexPoolId, balance, true);
    }

function _unstakeAllLp() internal override {
    Rewards(cvxCrvRewards).withdrawAllAndUnwrap(isClaimRewards);
}

function _unstakeLp(uint256 _amount) internal override {
    if (_amount != 0) {
        Rewards(cvxCrvRewards).withdrawAndUnwrap(_amount, false);
```

```
        }
    }
```

Recommendation

Always check the return values from external contract calls.

Status

All issues were resolved on commits

- `5afd3e2ef4b5e4ff6da779af2a78ae25b86ddcb5`
- `e2245a9daa86ae9fa49cae3c4410313c0c0dedc4`.

VSP-40	External protocols security incident could result in funds drained from strategies

Total Risk	Impact	Location
Low	Low	`CompoundXYStrategy.sol` `Earn.sol`
Fixed	Likelihood	
✔	Low	

Description

Vesper `CompoundXYStrategy` contract funds could be drained if any of the external protocols they interact with is compromised.

The `CompoundXYStrategy` trusts all its funds to the third party smart contracts by setting up an unlimited allowance as observed in the following code that takes place when a new borrow token is setup in replacement of a previous one in the `updateBorrowCToken` function:

```
        // Approve new borrowCToken
        IERC20(borrowToken).safeApprove(_newBorrowCToken, type(uint256).max);
```

This differs from how all the rest of the tokens in possession of the strategy are handled and the borrow token itself in the `_approveToken` function:

```
    /// @notice Approve all required tokens
    function _approveToken(uint256 _amount) internal virtual override {
        collateralToken.safeApprove(pool, _amount);
        collateralToken.safeApprove(address(supplyCToken), _amount);
        IERC20(borrowToken).safeApprove(address(borrowCToken), _amount);
        for (uint256 i = 0; i < swapManager.N_DEX(); i++) {
            IERC20(COMP).safeApprove(address(swapManager.ROUTERS(i)),
_amount);
            collateralToken.safeApprove(address(swapManager.ROUTERS(i)),
_amount);
        }
    }
```

Even though this is convenient from a gas utilization point of view, it introduces an undesirable explicit trust in the external smart contracts.

A similar scenario is observed in the `Earn.sol` contract:

```
    /// @dev Approves EarnDrip' Grow token to spend dripToken
    function approveGrowToken() external onlyKeeper {
        address _dripContract = IVesperPool(pool).poolRewards();
        address _growPool = IEarnDrip(_dripContract).growToken();
        // Checks that the Grow Pool supports dripToken as underlying
        require(address(IVesperPool(_growPool).token()) == dripToken,
"invalid-grow-pool");
        IERC20(dripToken).safeApprove(_growPool, MAX_UINT_VALUE);
    }
```

However, the risk is lower because the approved contract is owned by Vesper itself.

Recommendation

Consider only approving the exact amount required each time a transfer operation is performed in order to maximize the security of the funds handled by Vesper.

If the cost of doing so is considered prohibitive and/or not worthy when contrasted with the perceived risk of the external contracts being exploited, it is recommended to have an emergency response plan in place to facilitate a quick reaction in the unlikely event this happens.

Status

This issue is considered fixed as it has been deemed not relevant during Coinspect's latest analysis of Vesper's rationale and taking into consideration the extremely low risk involved, given that `growPool` is a trusted contract.

VSP-41	Lax access controls to reward token distribution configuration

Total Risk	Impact	Location
Medium	High	`EarnDrip.sol`

Fixed	Likelihood
✘	Low

Description

Any strategy in a pool can update reward distribution for any reward token in a pool. Even though an unlikely scenario, an ill intended strategy could increase (or decrease) the amount of any of the reward tokens being distributed by the pool.

The access checks in `notifyRewardAmount` function allows the pool governor or any active strategy to do that:

```
    /**
     * @dev Notify that reward is added.
     * Also updates reward rate and reward earning period.
     */
    function notifyRewardAmount(
        address _rewardToken,
        uint256 _rewardAmount,
        uint256 _rewardDuration
    ) external override {
        (bool isStrategy, , , , , , ) =
             IVesperPool(pool).strategy(msg.sender);
        require(
            msg.sender == IVesperPool(pool).governor() ||
                    (isRewardToken[_rewardToken] && isStrategy),
            "not-authorized"
        );
        super._notifyRewardAmount(_rewardToken, _rewardAmount,
_rewardDuration,
             IVesperPool(pool).totalSupply());
    }
```

Recommendation

Consider limiting the rewards distribution updates to the pool's Governor only.

Status

This is considered the expected behaviour and will not be fixed.

VSP-42	Broken accounting on special ERC20

Total Risk	Impact	Location
Medium	High	`contracts/pool/VFR.sol`

Fixed	Likelihood
✔	Low

Description

The `generateReport` function might report a wrong profit value. This can happen if tokens have fees on transfer.

Some `ERC20` tokens have a fee on transfer (like USDT though it is set to 0% now) which may lead to a wrong profit report on `VFRStablePool`.

```
[contracts/strategies/VFR.sol]
36  if (amountNeeded > 0) {
37      uint256 amountInBuffer = collateralToken.balanceOf(buffer);
38      uint256 amountReceived = amountInBuffer >= amountNeeded ? amountNeeded :
amountInBuffer;
39      IVFRBuffer(buffer).request(amountReceived);
40      _profit += amountReceived;
41  }
```

In line `39` the request might transfer less than the requested amount causing an error on line `40` when accounting the parameter sent as the increase of profits, instead of the actual received amount.

This will cause the generateReport function to call `reportEarning` with a wrong `_profit` value in `Strategy.sol:169`, being the actual profit lower. This might break some future requirements where it will expect that the pool balance to be greater or equal to the profit, reverting the `rebalance` call.

Recommendation

Check balances before and after the request when using tokens with fees to calculate the transferred value.

Status

This issue was fixed at commit `6908db15d4b240e8620cb68cb617c9819f87312a`.

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.