Orange Finance Audit Report

Apr 25, 2023

WATCHPUG

Table of Contents

Summary

This report has been prepared for Orange Finance Audit Report smart contract, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Overview

Project Summary

Project Name	**Orange Finance Audit Report**
Codebase	**https://github.com/orange-finance/alpha-contract**
Commit	**37659c1a5b2d091813b236a4a9dd4e75fb99bb97**
Language	**Solidity**

Audit Summary

Delivery Date	**Apr 25, 2023**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**7**

[WP-H1] Attacker can manipulate the price (to a lower value), mint and reverse the price and burn to steal funds from other share holders

High

Issue Description

https://github.com/orange-finance/alpha-contract/blob/d70731409d520555dab276594bd45eff7c43f82f/contracts/core/OrangeAlphaVault.sol#L745-L821

```
function deposit(
    uint256 _assets,
    address _receiver,
    uint256 _minShares
) external returns (uint256 shares_) {
    //validation
    if (_receiver != msg.sender) {
        revert(Errors.DEPOSIT_RECEIVER);
    }
    if (_assets == 0) revert(Errors.DEPOSIT_ZERO);
    if (deposits[_receiver].assets + _assets > depositCap(_receiver)) {
        revert(Errors.DEPOSIT_CAP_OVER);
    }
    deposits[_receiver].assets += _assets;
    if (totalDeposits + _assets > totalDepositCap) {
        revert(Errors.TOTAL_DEPOSIT_CAP_OVER);
    }
    totalDeposits += _assets;

    Ticks memory _ticks = _getTicksByStorage();

    //mint
    shares_ = _convertToShares(_assets, _ticks);
    if (_minShares > shares_) {
        revert(Errors.LESS_THAN_MIN_SHARES);
    }
    _mint(_receiver, shares_);
@@ 772,820 @@
}
```

https://github.com/orange-finance/alpha-contract/blob/96be6d17b9185aa8c499332364aff8f6b9f07dee/contracts/core/OrangeAlphaVault.sol#L229-L250

```
function _totalAssets(Ticks memory _ticks) internal view returns (uint256) {
    UnderlyingAssets memory _underlyingAssets = _getUnderlyingBalances(
        _ticks
    );

    // Aave positions
    uint256 amount0Debt = debtToken0.balanceOf(address(this));
    uint256 amount1Supply = aToken1.balanceOf(address(this));

    return
        _alignTotalAsset(
            _ticks,
            _underlyingAssets.amount0Current +
                _underlyingAssets.accruedFees0 +
                _underlyingAssets.amount0Balance,
            _underlyingAssets.amount1Current +
                _underlyingAssets.accruedFees1 +
                _underlyingAssets.amount1Balance,
            amount0Debt,
            amount1Supply
        );
}
```

https://github.com/orange-finance/alpha-contract/blob/d70731409d520555dab276594bd45eff7c43f82f/contracts/core/OrangeAlphaVault.sol#L357-L392

```
function _getUnderlyingBalances(Ticks memory _ticks)
    internal
    view
    returns (UnderlyingAssets memory underlyingAssets)
{
    (
        uint128 liquidity,
        uint256 feeGrowthInside0Last,
        uint256 feeGrowthInside1Last,
        uint128 tokensOwed0,
        uint128 tokensOwed1
    ) = pool.positions(_getPositionID(_ticks.lowerTick, _ticks.upperTick));

    // compute current holdings from liquidity
    if (liquidity > 0) {
        (
            underlyingAssets.amount0Current,
            underlyingAssets.amount1Current
        ) = LiquidityAmounts.getAmountsForLiquidity(
            _ticks.sqrtRatioX96,
            _ticks.lowerTick.getSqrtRatioAtTick(),
            _ticks.upperTick.getSqrtRatioAtTick(),
            liquidity
        );
    }

    underlyingAssets.accruedFees0 =
        _computeFeesEarned(true, feeGrowthInside0Last, liquidity, _ticks) +
        uint256(tokensOwed0);
    underlyingAssets.accruedFees1 =
        _computeFeesEarned(false, feeGrowthInside1Last, liquidity, _ticks) +
        uint256(tokensOwed1);

    underlyingAssets.amount0Balance = token0.balanceOf(address(this));
    underlyingAssets.amount1Balance = token1.balanceOf(address(this));
}
```

`_totalAssets()` will be used to calculate the price per share in `mint()` .

However, as the `ticks.sqrtRatioX96` can be manipulated, there is a chance for the attacker to manipulate the price of ETH to a lower value, mint and reverse the price and burn to steal funds from other share holders.

PoC

Notice: This is NOT a traditional MEV sandwich attack vector, therefore it CAN NOT be prevented by introducing a proper slippage control parameter.

Given:

- market price of ETH: 2000 USDC
- total vault holdings: 1000 ETH + 1 USDC, worth $2M
- totalShares: 100

The attacker can:

1. Manipulate the price of the pool by swapping a huge amount of ETH to USDC, crashing the price from `2000` to `1000` .
2. `deposit()` with 10k USDC. As the price is now manipulated, `_totalAssets()` is worth only $1M, therefore, the attacker will get `2` shares.
3. Restore the price by swapping USDC back to ETH;
4. `redeem()` the `2` shares received from step 2, as the price has been restored, the `_totalAssets()` is about $2M now, therefore the attacker will get about 2000 USDC.

Recommendation

Consider changing the `deposit()` function to always add liquidity and hedge position proportionally to the current holdings. And pull the amount of USDC needed from the user's wallet directly.

Status

✓ Fixed

[WP-H2] `rebalance()` should add collateral when needed

High

Issue Description

https://github.com/orange-finance/alpha-contract/blob/d70731409d520555dab276594bd45eff7c43f82f/contracts/core/OrangeAlphaVault.sol#L947-L1036

```
 947  function rebalance(int24 _newLowerTick, int24 _newUpperTick)
 948          external
 949          onlyOwner
 950      {
      @@ 951,978 @@
 979          //calculate repay or borrow amount
 980          (, uint256 _newBorrow) = _computeSupplyAndBorrow(
 981              _totalAssets(_ticks),
 982              _ticks
 983          );
 984
 985          // 4. Swap
 986          // 5. Repay or borrow (if swapping from ETH to USDC, do borrow)
      @@ 987,1007 @@
1008          } else {
1009              //borrow
1010              aave.borrow(
1011                  address(token0),
1012                  _newBorrow - _debtBalance,
1013                  2,
1014                  0,
1015                  address(this)
1016              );
1017          }
1018
1019          // 6. Add liquidity
1020          uint256 reinvest0 = token0.balanceOf(address(this));
1021          uint256 reinvest1 = token1.balanceOf(address(this));
1022          _swapAndAddLiquidity(reinvest0, reinvest1, _ticks);
1023
```

```
        (uint128 newLiquidity, , , , ) = pool.positions(
            _getPositionID(_ticks.lowerTick, _ticks.upperTick)
        );
        if (newLiquidity == 0) {
            revert(Errors.NEW_LIQUIDITY_ZERO);
        }

        emit Rebalance(_newLowerTick, _newUpperTick, liquidity, newLiquidity);
        _emitAction(3, _ticks);

        //reset stoplossed
        stoplossed = false;
    }
```

When `_newBorrow` is greater than `_debtBalance` , `rebalance()` will borrow more to increase the hedge position.

It should add collateral at the same time, if needed.

Otherwise, the health factor on Aave can be low, putting the whole hedge position at risk of liquidation.

Status

✓ Fixed

[WP-H3] `_swapAndAddLiquidity()` can cause loss to the other shareholders when the market is volatile

High

Issue Description

https://github.com/orange-finance/alpha-contract/blob/d70731409d520555dab276594bd45eff7c43f82f/contracts/core/OrangeAlphaVault.sol#L796-L803

```
// 4. Swap from USDC to ETH (if necessary)
// 5. Add liquidity
uint256 _addingUsdc = _assets - _supply;
(
    _liquidity,
    _amountDeposited0,
    _amountDeposited1
) = _swapAndAddLiquidity(_borrow, _addingUsdc, _ticks);
```

_ `swapAndAddLiquidity()` in `deposit()` is using the vault's public money to swap, so the slippage is at the cost of all shareholders.

Attacker can exploit this with a sandwich attack on `deposit()` .

Recommendation

See the Recommendation on [WP-H1].

Status

✓ Fixed

[WP-H4] A malicious early user/attacker can manipulate the pricePerShare to take an unfair share of future users' deposits

High

Issue Description

https://github.com/orange-finance/alpha-contract/blob/96be6d17b9185aa8c499332364aff8f6b9f07dee/contracts/core/OrangeAlphaVault.sol#L689-L757

```
function deposit(uint256 _assets, address _receiver)
    external
    returns (uint256 shares_)
{
    //validation
    if (_receiver != msg.sender) {
        revert InvalidDepositReceiver();
    }
    if (_assets == 0) revert InvalidDepositZero();
    if (deposits[_receiver].assets + _assets > depositCap(_receiver)) {
        revert InvalidDepositCapOver();
    }
    deposits[_receiver].assets += _assets;
    if (totalDeposits + _assets > totalDepositCap) {
        revert InvalidTotalDepositCapOver();
    }
    totalDeposits += _assets;

    Ticks memory _ticks = _getTicksByStorage();

    //mint
    shares_ = _convertToShares(_assets, _ticks);
    _mint(_receiver, shares_);
@@ 712,756 @@
}
```

https://github.com/orange-finance/alpha-contract/blob/96be6d17b9185aa8c499332364aff8f6b9f07dee/contracts/core/OrangeAlphaVault.sol#L253-L263

```solidity
function _convertToShares(uint256 _assets, Ticks memory _ticks)
    public
    view
    returns (uint256 shares)
{
    uint256 supply = totalSupply(); // Saves an extra SLOAD if totalSupply is non-zero.
    return
        supply == 0
            ? _assets
            : _assets.mulDiv(supply, _totalAssets(_ticks));
}
```

The first minter can `deposit()` , and then withdraw all but a small amount (eg, `199` wei) of the deposit to infalte the pps of the vault.

- `deposit()` `1e18 wei` and get `1e18 wei` of shares;
- `redeem()` `1e18 - 199 wei` of shares

Then the attacker can send `100e18 - 199` of tokens and inflate the price per share to `100e18 * 1e18 / 199` .

As a result, the future user who deposits `1e18` will only receive `1e18 * 199 / 100e18 = 1 wei` of shares.

They will immediately lose `0.495e18` or half of their deposits if they `redeem()` right after the `deposit()` .

Recommendation

Consider requiring a minimal amount of shares to be minted for the first minter, and send a portion of the initial mints as a reserve to the DAO so that the pricePerShare can be more resistant to manipulation.

Status

✓ Fixed

[WP-M5] Lack of slippage control for `rebalance()`

Medium

Issue Description

https://github.com/orange-finance/alpha-contract/blob/96be6d17b9185aa8c499332364aff8f6b9f07dee/contracts/core/OrangeAlphaVault.sol#L880-L883

```
    function rebalance(int24 _newLowerTick, int24 _newUpperTick)
        external
        onlyOwner
    {
```

There is no slippage control in the `rebalance()` function.

This means that a sudden market movement or an intentional frontrun price manipulation may result in a different output for the caller (the manager).

Specifically, a different `newLiquidity` as the result of the `rebalance()` .

Status

✓ Fixed

[WP-I6] `OrangeAlphaVault.constructor()` depends on the order of `(token0, token1)` being `(weth, usdc)` , which is not the case on Ethereum and Polygon

Informational

Issue Description

https://github.com/orange-finance/alpha-contract/blob/96be6d17b9185aa8c499332364aff8f6b9f07dee/contracts/core/OrangeAlphaVault.sol#L48-L52

```
    IERC20 public token0; //weth
    IERC20 public token1; //usdc
    IAaveV3Pool public aave;
    IERC20 public debtToken0; //weth
    IERC20 public aToken1; //usdc
```

https://github.com/orange-finance/alpha-contract/blob/96be6d17b9185aa8c499332364aff8f6b9f07dee/contracts/core/OrangeAlphaVault.sol#L64-L104

```
    constructor(
        string memory _name,
        string memory _symbol,
        address _pool,
        address _aave,
        int24 _lowerTick,
        int24 _upperTick
    ) ERC20(_name, _symbol) {
        // setting adresses and approving
        pool = IUniswapV3Pool(_pool);
        token0 = IERC20(pool.token0());
        token1 = IERC20(pool.token1());
        token0.safeApprove(_pool, type(uint256).max);
        token1.safeApprove(_pool, type(uint256).max);

        aave = IAaveV3Pool(_aave);
        token0.safeApprove(_aave, type(uint256).max);
        token1.safeApprove(_aave, type(uint256).max);
        DataTypes.ReserveData memory reserveDataToken0 = aave.getReserveData(
            address(token0)
        );
        debtToken0 = IERC20(reserveDataToken0.variableDebtTokenAddress);
        DataTypes.ReserveData memory reserveDataToken1 = aave.getReserveData(
            address(token1)
        );
        aToken1 = IERC20(reserveDataToken1.aTokenAddress);
        //this decimal is same as token1's decimal
        _decimal = IERC20Decimals(address(token1)).decimals();

        // these variables can be udpated by the manager
        _depositCap = 1_000_000 * 1e6;
        totalDepositCap = 1_000_000 * 1e6;
        slippageBPS = 500; // default: 5% slippage
        slippageInterval = 5 minutes;
        maxLtv = 8000; //80%

        //setting ticks
        _validateTicks(_lowerTick, _upperTick);
        lowerTick = _lowerTick;
        upperTick = _upperTick;
    }
```

- USDC / ETH 0.05% on Ethereum: https://etherscan.io/address/0x88e6a0c2ddd26feeb64f039a2c41296fcb3f5640#readContract
  - token0 ⊠ USDC
- USDC / WETH 0.05% on Polygon: https://polygonscan.com/address/0x45dda9cb7c25131df268515131f647d726f50608#readContract
  - token0 ⊠ USDC

Status

ⓘ Acknowledged

[WP-G7] Calling `OracleLibrary.getQuoteAtTick()` only when needed

Gas

Issue Description

https://github.com/orange-finance/alpha-contract/blob/d70731409d520555dab276594bd45eff7c43f82f/contracts/core/OrangeAlphaVault.sol#L319-L348

```
    function _alignTotalAsset(
        Ticks memory _ticks,
        uint256 amount0Current,
        uint256 amount1Current,
        uint256 amount0Debt,
        uint256 amount1Supply
    ) internal view returns (uint256 totalAlignedAssets) {
        if (amount0Current < amount0Debt) {
            uint256 amount0deducted = amount0Debt - amount0Current;
            amount0deducted = OracleLibrary.getQuoteAtTick(
                _ticks.currentTick,
                uint128(amount0deducted),
                address(token0),
                address(token1)
            );
            totalAlignedAssets =
                amount1Current +
                amount1Supply -
                amount0deducted;
        } else {
            uint256 amount0Added = amount0Current - amount0Debt;
            amount0Added = OracleLibrary.getQuoteAtTick(
                _ticks.currentTick,
                uint128(amount0Added),
                address(token0),
                address(token1)
            );
            totalAlignedAssets = amount1Current + amount1Supply + amount0Added;
        }
    }
```

Recommendation

```
    function _alignTotalAsset(
        Ticks memory _ticks,
        uint256 amount0Current,
        uint256 amount1Current,
        uint256 amount0Debt,
        uint256 amount1Supply
    ) internal view returns (uint256 totalAlignedAssets) {
        if (amount0Current < amount0Debt) {
            uint256 amount0deducted = amount0Debt - amount0Current;
            amount0deducted = OracleLibrary.getQuoteAtTick(
                _ticks.currentTick,
                uint128(amount0deducted),
                address(token0),
                address(token1)
            );
            totalAlignedAssets =
                amount1Current +
                amount1Supply -
                amount0deducted;
        } else {
            uint256 amount0Added = amount0Current - amount0Debt;
            if (amount0Added > 0)
                amount0Added = OracleLibrary.getQuoteAtTick(
                    _ticks.currentTick,
                    uint128(amount0Added),
                    address(token0),
                    address(token1)
                );
            totalAlignedAssets = amount1Current + amount1Supply + amount0Added;
        }
    }
```

Status

✓ Fixed

Appendix

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by WatchPug; however, WatchPug does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Disclaimer