y

BT block

Security Code Review for YOP Protocol EVM

Pluto Digital

February 2022

Version 2.0

Presented by:
BTblock LLC

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

Executive Summary

Overview

Pluto Digital engaged BTblock LLC to perform a Security Code Review for YOP Protocol EVM.

The assessment was conducted remotely by the BTblock Security Team. Testing took place on January 25 - February 10, 2022, and focused on the following objectives:

- Provide the customer with an assessment of their overall security posture and any risks discovered within the environment during the engagement.
- To provide a professional opinion on the security measures' maturity, adequacy, and efficiency.
- To identify potential issues and include improvement recommendations based on the result of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the BTblock Security Teams took to identify and validate each issue, as well as any applicable recommendations for remediation.

Key Findings

The following are the major themes and issues identified during the testing period. Within the findings section, these, along with other items, should be prioritized for remediation to reduce the risk they pose.

During the test, the following positive observations were noted regarding the scope of the engagement:

- The team was very supportive and open to discussing the design choices made

We conclude that the reviewed code implements the documented functionality based on formal verification.

Scope and Rules of Engagement

BTblock performed a Security Code Review for YOP Protocol EVM. The following table documents the targets in scope for the engagement. No other systems or resources were in scope for this assessment.

The source code was supplied through a private repository at https://github.com/plutodigital/yop-protocol-evm with the commit hash e610319fc3ad3650781b0a17e85951bf078008b3.

Files included in the code review

```
yop-protocol-evm/
├── contracts/
│   ├── access/
│   │   ├── AccessControlManager.sol
│   │   ├── AllowAnyAccessControl.sol
│   │   ├── AllowListAccessControl.sol
│   │   ├── ERC1155AccessControl.sol
│   │   └── PerVaultGatekeeper.sol
│   ├── fees/
│   │   └── FeeCollection.sol
│   ├── interfaces/
│   │   ├── convex/
│   │   │   ├── IConvexDeposit.sol
│   │   │   └── IConvexRewards.sol
│   │   ├── curve/
│   │   │   ├── ICurveAddressProvider.sol
│   │   │   ├── ICurveDeposit.sol
│   │   │   ├── ICurveGauge.sol
│   │   │   ├── ICurveMinter.sol
│   │   │   └── ICurveRegistry.sol
│   │   ├── roles/
│   │   │   └── IGatekeeperable.sol
│   │   ├── sushiswap/
│   │   │   └── IUniswapV2Router.sol
│   │   ├── IAccessControl.sol
│   │   ├── IAccessControlManager.sol
│   │   ├── ICustomHealthCheck.sol
│   │   ├── IFeeCollection.sol
│   │   ├── IHealthCheck.sol
│   │   ├── IStrategy.sol
│   │   ├── IVault.sol
│   │   ├── IVaultStrategyDataStore.sol
│   │   ├── IWeth.sol
│   │   └── IYOPRewards.sol
│   ├── libraries/
│   │   └── ConvertUtils.sol
│   ├── mocks/
│   │   ├── BasePauseableUpgradeableMock.sol
│   │   ├── BaseStrategyMock.sol
│   │   ├── BaseVaultMock.sol
│   │   ├── ConvexBtcStrategyMock.sol
│   │   ├── ConvexEthStrategyMock.sol
│   │   ├── ConvexStableStrategyMock.sol
│   │   ├── CurveBtcStrategyMock.sol
│   │   ├── CurveEthStrategyMock.sol
│   │   ├── CurveStableStrategyMock.sol
│   │   ├── CustomHealthCheckMock.sol
│   │   ├── HealthCheckMock.sol
│   │   ├── SingleAssetVaultV2Mock.sol
│   │   ├── StakingMock.sol
│   │   ├── StakingV1Mock.sol
│   │   ├── StrategyMock.sol
│   │   ├── TestnetStrategyMock.sol
```

```
│   │   ├── TokenMock.sol
│   │   ├── YOPRewardsMock.sol
│   │   ├── YOPTokenMock.sol
│   │   └── YopERC1155Mock.sol
│   ├── rewards/
│   │   └── YOPRewards.sol
│   ├── security/
│   │   └── BasePauseableUpgradeable.sol
│   ├── staking/
│   │   └── Staking.sol
│   ├── strategies/
│   │   ├── BaseStrategy.sol
│   │   ├── ConvexBase.sol
│   │   ├── ConvexBtc.sol
│   │   ├── ConvexEth.sol
│   │   ├── ConvexStable.sol
│   │   ├── CurveBase.sol
│   │   ├── CurveBtc.sol
│   │   ├── CurveEth.sol
│   │   └── CurveStable.sol
│   └── vaults/
│       ├── roles/
│       │   ├── Gatekeeperable.sol
│       │   ├── Governable.sol
│       │   └── Manageable.sol
│       ├── BaseVault.sol
│       ├── CommonHealthCheck.sol
│       ├── SingleAssetVault.sol
│       ├── SingleAssetVaultBase.sol
│       ├── VaultDataStorage.sol
│       ├── VaultMetaDataStore.sol
│       └── VaultStrategyDataStore.sol
├── deployment-config/
│   ├── config-example.yaml
│   ├── config-test-rinkeby.yaml
│   └── vaults.ts
├── deployments/
│   ├── rinkeby.json
│   └── test-rinkeby.json
├── flat/
│   ├── AccessControlManager_flat.sol
│   ├── Inbox_flat.sol
│   └── SingleAssetVault_flat.sol
├── scripts/
│   ├── gnosis/
│   │   ├── propose-txn.ts
│   │   └── safe-create.ts
│   ├── lib/
│   │   ├── AccessControlManagerDeployment.ts
│   │   ├── AllowAnyAccessControlDeployment.ts
│   │   ├── AllowlistAccessControlDeployment.ts
│   │   ├── ContractDeployment.ts
│   │   ├── ConvexStrategyDeployment.ts
```

```
│   │   ├── CurveStrategyDeployment.ts
│   │   ├── ERC1155AccessControlDeployment.ts
│   │   ├── Executor.ts
│   │   ├── FeeCollectionDeployment.ts
│   │   ├── MockStrategyDeployment.ts
│   │   ├── StakingDeployment.ts
│   │   ├── VaultDeployment.ts
│   │   ├── VaultStrategyDataStoreDeployment.ts
│   │   └── YopRewardDeployment.ts
│   ├── README.md
│   ├── deploy-all.ts
│   ├── deploy-by-config.ts
│   ├── deploy-contract.ts
│   ├── deploy-mock.ts
│   ├── propose-upgrade.ts
│   ├── util.ts
│   └── verify.ts
├── tasks/
│   ├── fork/
│   │   ├── fundAccounts.ts
│   │   ├── impersonateAccounts.ts
│   │   └── reset.ts
│   ├── gnosis/
│   │   ├── propose-txn.ts
│   │   └── safe-create.ts
│   ├── rewards/
│   │   └── approveRewardsContract.ts
│   └── index.ts
├── README.md
├── commitlint.config.js
├── constants.ts
├── hardhat.config.ts
├── package-lock.json
├── package.json
├── tenderly.yaml
└── tsconfig.json
```

Table 1: Scope

Technical Analyses and Findings

During the Security Code Review for YOP Protocol EVM, we discovered:

- One finding with a MEDIUM severity rating.
- Three findings with a LOW severity rating.
- One finding with an INFORMATIONAL severity rating.

The following chart displays the findings by severity.

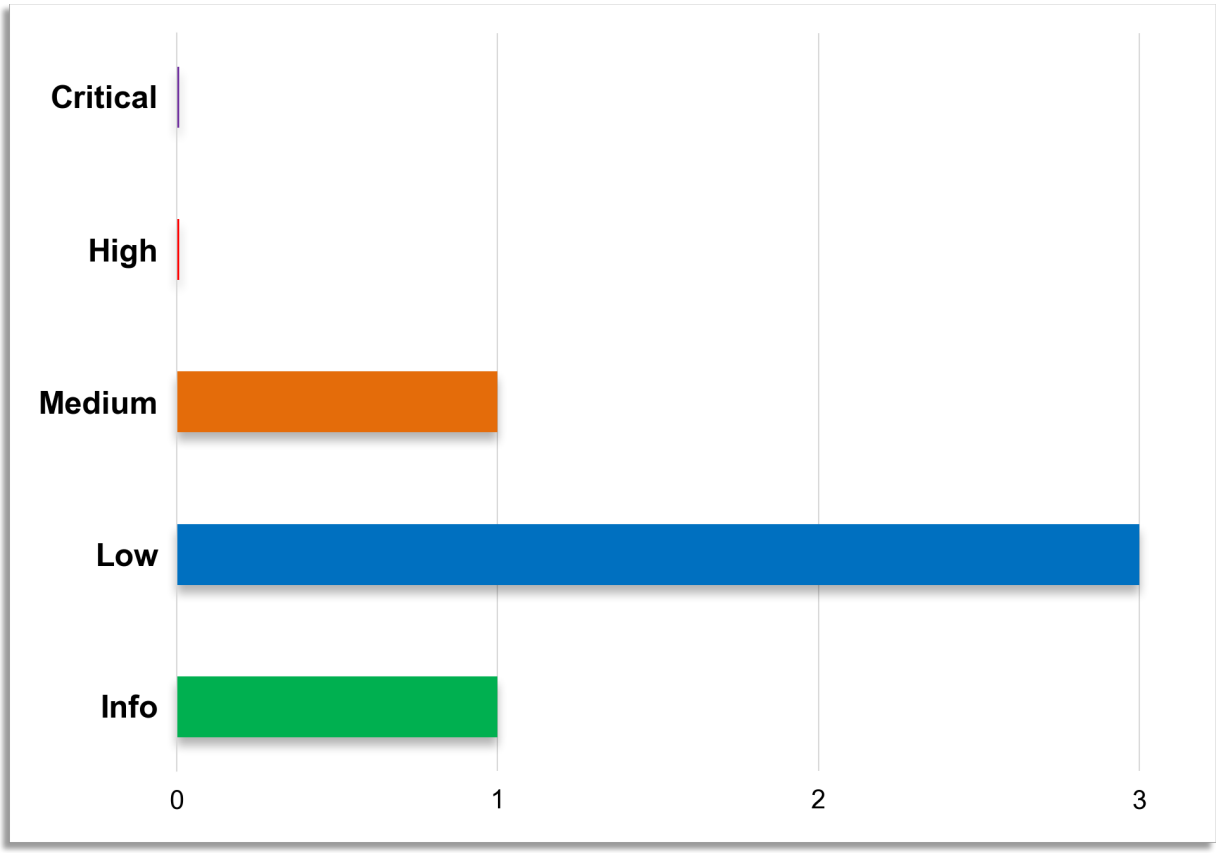

Figure 1: Findings by Severity

Findings

The *Findings* section provides detailed information on each finding, including discovery methods, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

#	Severity	Description
KS-YOP-EVM-01	Medium	The contract does not protect the initialize function
KS-YOP-EVM-02	Low	It is possible to add a strategy smart contract that is not an inherited strategy interface
KS-YOP-EVM-03	Low	It is possible to add a vault smart contract that is not meet the vault interface
KS-YOP-EVM-04	Low	IAccessControl is re-used
KS-YOP-EVM-05	Informational	Not all smart contract code is covered with unit tests

Table 2: Findings Overview

Technical Analyses

Conclusion

Based on formal verification, we conclude that the code implements the documented functionality to the extent of the reviewed code.

Technical Findings

General Observations

The Yield Optimization Platform (YOP) enables users to interact with the DeFi protocols across the top blockchains. Initial strategy development will focus on a number of the top DeFi Protocols on the Ethereum blockchain. These include Aave, Convex, Curve, Uniswap, Sushiswap, IndexCoup, and LIDO. The platform is developed with `Solidity` and will be placed in the `Ethereum` network

Summary of strengths

- The code and project files are well structured, making them easy to read and maintain. The code is self-explanatory. The naming policy makes instructions understandable. All public functions are documented in comments.
- Well-designed and clearly defined smart contract access rights.
- The repository contains documentation and a general architecture observation.
- The contracts perform only the declared functionality.
- The main execution scenario has been covered with unit tests.
- Verification errors have a custom explanation.
- The contracts are developed using the up-to-date compiler.

Summary of discovered vulnerabilities

During the assessment, 1 informational, 3 low, and 1 medium-level issue was discovered. The medium issue could lead to loss of contract control. The Low-level issues could lead to temporary stopping the platform.

At the re-review, all issues were remediated, except one based on other means was accepted at risk by Pluto Digital.

The contract does not protect the initialize function

Finding ID: KS-YOP-EVM-01
Severity: Medium
Status: Accepted at risk by the client.

Description

The contract FeeCollection does not protect initialize function

Proof of issue

Filename: contracts/fees/FeeCollection.sol
Line number: 86

```
      function initialize(
        address _governance,
        address _gatekeeper,
        address _protocolWallet,
        address _vaultStrategyDataStore,
        uint16 _defaultVaultCreatorFeeRatio,
        uint16 _defaultStrategyProposerFeeRatio,
        uint16 _defaultStrategyDeveloperFeeRatio
    ) external initializer {
        __FeeCollection_init(
            _governance,
            _gatekeeper,
            _protocolWallet,
            _vaultStrategyDataStore,
            _defaultVaultCreatorFeeRatio,
            _defaultStrategyProposerFeeRatio,
            _defaultStrategyDeveloperFeeRatio
        );
  }
```

Severity and Impact summary

Some users can call `the initialize function before the contract owner, leading to a loss of` contract control.

Recommendation

Add a constructor to ensure that `initialize` cannot be called from outside.

It is possible to add a strategy smart contract that is not inherited from the strategy interface

Finding ID: KS-YOP-EVM-02
Severity: Low
Status: Remediated

Description

In smart contract `VaultStrategyDataStore` there is a function `addStrategy` that takes strategy address as one of the parameters and adds it into specified vault strategy storage. Strategy storage is used for funds withdrawal and reports. The problem is that `addStrategy` does not check if the smart contract with strategy address corresponds to the strategy interface.

Proof of issue

File name: contracts/vaults/VaultStrategyDataStore.sol
Line number: 231

```
  function addStrategy(
    address _vault,
    address _strategy,
    uint256 _debtRatio,
    uint256 _minDebtPerHarvest,
    uint256 _maxDebtPerHarvest,
    uint256 _performanceFee
  ) external {
    _onlyGovernanceOrVaultManager(_vault);
    require(_strategy != address(0), "strategy address is not valid");
    _initConfigsIfNeeded(_vault);
    require(configs[_vault].withdrawQueue.length <
MAX_STRATEGIES_PER_VAULT, "too many strategies");
    require(strategies[_vault][_strategy].activation == 0, "strategy
already added");
    if (IStrategy(_strategy).vault() != address(0)) {
      require(IStrategy(_strategy).vault() == _vault, "wrong vault");
    }
    require(_minDebtPerHarvest <= _maxDebtPerHarvest, "invalid
minDebtPerHarvest value");
    require(
      configs[_vault].totalDebtRatio + _debtRatio <=
configs[_vault].maxTotalDebtRatio,
      "total debtRatio over limit"
    );
    require(_performanceFee <= MAX_BASIS_POINTS / 2, "invalid performance
fee");
```

```
    ...
}
```

Severity and Impact summary

If the smart contract that does not inherit strategy interface is added to `WithdrawQueue` of `VaultStrategyDataStore` contract, `the withdraw function of the SingleAssetVault smart contract will fail when the user tries to withdraw` his funds. In this case, the admin will be forced to pause the vault and remove the non-strategy address from the withdrawQueue, which leads to system downtime and additional load on support.

Recommendation

It is recommended to add logic that checks the strategy interface in the `addStrategy`.

It is possible to add a vault smart contract that does not meet the vault interface

Finding ID: KS-YOP-EVM-03
Severity: Low
Status: Remediated

Description

A function `setPerVaultRewardsWeight` receives an array of addresses of vaults in the first parameter. Users with role Governance could call this function. The function `setPerVaultRewardsWeight` does not check that received contracts meet the vault interface. It only cast the addresses to the `ERC20MetadataUpgradeable` interface. It means that it is accidentally possible to send one of the non-vault contract addresses with `ERC20MetadataUpgradeable` interface.

Proof of issue

File name: contracts/rewards/YOPRewards.sol
Line number: 221

```
   function setPerVaultRewardsWeight(address[] calldata _vaults, uint256[]
calldata _weights) external onlyGovernance {
    require(_vaults.length > 0, "!vaults");
    require(_vaults.length == _weights.length, "!sameLength");
    ...
    emit VaultRewardWeightUpdated(_vaults, _weights);
  }
```

Severity and Impact summary

If passing a non-vault contract to the `setPerVaultRewardsWeight` function, the overall reward will be decreased, and other vaults will have their rewards rate decreased. Decreased rate will decrease rewards for each vault in the `_calculatePoolState` method.

Recommendation

Add check for _vaults

References

https://gitlab.com/btblock-cybersec/yop-protocol-evm/-/blob/main/contracts/rewards/YOPRewards.sol

IAccessControl is re-used

Finding ID: KS-YOP-EVM-04
Severity: Low
Status: Remediated

Description

The interface's name IAccessControl is used in the openzeppelin and the yop protocol.

Proof of issue

File name: node_modules/@openzeppelin/contracts/access/IAccessControl.sol
Line number: 49

```
    abstract contract AccessControl is Context, IAccessControl, ERC165 {
        ...
    }
```

File name: contracts/interfaces/IAccessControl.sol
Line number: 4

```
    interface IAccessControl {
        function hasAccess(address _user, address _vault) external view
returns (bool);
    }
}
```

Severity and Impact summary

If a codebase has two similar contracts, the compilation artifacts will not contain duplicate names. The exploited issue scenario: Bob's truffle codebase has two contracts with the same name. When truffle compile runs, only one of the two contracts will generate artifacts in build/contracts. As a result, the second contract cannot be analyzed.

Recommendation

Rename interface in contracts/interfaces/IAccessControl.sol

References

https://github.com/crytic/slither/wiki/Detector-Documentation#name-reused
https://github.com/plutodigital/yop-protocol-evm/contracts/interfaces/IAccessControl.sol

Not all smart contract code is covered with unit tests

Finding ID: KS-YOP-EVM-05
Severity: Informational

Description

Unit tests are an essential part of innovative contract development. They help find bugs and security issues in the code that the developer misses before deploying the contract to the blockchain. We found that not all intelligent contract code is covered with unit tests.

Proof of issue

The percentage of unit tests coverage for each contract was evaluated during the audit. The results are presented in the table below.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line nums
access/	100	90.43	100	100	
AccessControlManager.sol	100	100	100	100	
AllowAnyAccessControl.sol	100	100	100	100	
AllowListAccessControl.sol	100	94.44	100	100	
ERC1155AccessControl.sol	100	81.82	100	100	
PerVaultGatekeeper.sol	100	100	100	100	
fees/	96.51	75	100	96.51	
FeeCollection.sol	96.51	75	100	96.51	292,302,310
libraries/	100	100	100	100	
ConvertUtils.sol	100	100	100	100	
rewards/	100	88.75	100	100	
YOPRewards.sol	100	88.75	100	100	
security/	100	100	100	100	
BasePauseableUpgradeable.sol	100	100	100	100	
staking/	100	100	100	100	
Staking.sol	100	100	100	100	
strategies/	94.46	79.82	92.54	94.13	
BaseStrategy.sol	98.96	86.96	96.67	97.87	127,128
ConvexBase.sol	95.45	57.14	90	95.45	73,115
ConvexBtc.sol	100	100	100	100	
ConvexEth.sol	100	100	90	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line nums
ConvexStable.sol	100	100	90.91	100	
CurveBase.sol	92.19	79.41	93.33	92.06	… 302,303,343
CurveBtc.sol	100	100	100	100	
CurveEth.sol	100	100	100	100	
CurveStable.sol	85.19	60	76.47	85.19	… 134,136,137
vaults/	96.93	94.29	95.45	96.57	
BaseVault.sol	72.22	83.33	76.47	72.22	… 153,154,155
CommonHealthCheck.sol	100	100	100	100	
SingleAssetVault.sol	98.63	85.71	96	97.28	371,380,480,517
SingleAssetVaultBase.sol	96.83	81.25	100	96.77	156,164
VaultDataStorage.sol	100	100	100	100	
VaultMetaDataStore.sol	96.49	100	94.74	96.49	112,113
VaultStrategyDataStore.sol	100	99.21	100	100	
vaults/roles/	100	100	100	100	
Gatekeeperable.sol	100	100	100	100	
Governable.sol	100	100	100	100	
All files	97.25	90.03	96.19	97.06	

Stmts has each executable line executed in the source file, including imports, class, and function definitions.
Branch - has each branch of each control structure (if case statements, for example) been executed.
Funcs - has each function in the program been called.
Line - has each executable line in the source file been executed.

Severity and Impact summary

Not having full code coverage with unit tests can lead to vulnerabilities or developer bugs that do not reveal before the contract's deployment.

Recommendation

It is recommended to cover 100% of the code with the unit tests.