HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Pluto Digital, YOP protocol
Date: February 17th, 2022

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for Pluto Digital, YOP protocol.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Vault; Staking
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/plutodigital/yop-protocol-evm
Commit	63a94637eaa423a111cd3d98b5875f8c32fa6182
Technical Documentation	YES
JS tests	YES
Website	https://plutodigital.com/ https://yop.finance/
Timeline	18 JANUARY 2022 - 17 FEBRUARY 2022
Changelog	10 FEBRUARY 2022 - Initial Audit 17 FEBRUARY 2022 - Second Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Pluto Digital, YOP protocol (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between January 18th, 2022 - February 17th, 2022.

Scope

The scope of the project is smart contracts in the repository:

Repository:
https://github.com/plutodigital/yop-protocol-evm
Commit:
63a94637eaa423a111cd3d98b5875f8c32fa6182
Technical Documentation: Yes
1. Whitepaper:
https://cdn.yop.finance/wp-content/uploads/2022/01/26140053/YOP_Whitepaper_final.pdf
2. Tech docs
https://github.com/plutodigital/yop-protocol-evm/tree/main/docs

JS tests: Yes
- https://github.com/plutodigital/yop-protocol-evm/tree/main/test

Contracts:
access/AccessControlManager.sol
access/AllowAnyAccessControl.sol
access/AllowListAccessControl.sol
access/ERC1155AccessControl.sol
access/PerVaultGatekeeper.sol
fees/FeeCollection.sol
libraries/ConvertUtils.sol
rewards/YOPRewards.sol
security/BasePauseableUpgradeable.sol
staking/Staking.sol
strategies/BaseStrategy.sol
strategies/ConvexBase.sol
strategies/ConvexBtc.sol
strategies/ConvexEth.sol
strategies/ConvexStable.sol
strategies/CurveBase.sol
strategies/CurveBtc.sol
strategies/CurveEth.sol
strategies/CurveStable.sol
vaults/roles/Gatekeeperable.sol
vaults/roles/Governable.sol
vaults/roles/Manageable.sol
vaults/BaseVault.sol
vaults/CommonHealthCheck.sol
vaults/SingleAssetVault.sol
vaults/SingleAssetVaultBase.sol
vaults/VaultDataStorage.sol
vaults/VaultMetaDataStore.sol

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

vaults/VaultStrategyDataStore.sol

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **1** medium and **6** low severity issues.

After the second review security engineers found that an additional interface support check was added to the adding new strategies, flash loan possibility was removed by putting a restriction to the user to transfer any tokens in the same block when tokens are minted and low issues were also addressed. Therefore security engineers didn’t find any security issues with the updated code.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Audit overview

Critical

No critical issues were found.

High

No high severity issues were found.

Medium

Tokens could be used for Flash Loan Attack

While the deposit function mints ERC20 tokens as shares in exchange for the depositing tokens, in the case both tokens are trading on the DEXes, this could be used in the Flash Loan Attack.

Contracts: SingleAssetVault.sol

Function: deposit, withdraw

Recommendation: Consider adding vesting in at least 1 block to the given ERC20 token, so it couldn't be used for any transactions within the same block.

Status: Fixed.

Low

1. Excess "approve" calls

Switching DEX from Uni to Sushi will make it to approve assets spending for DEX each time, even if it's already approved.

Contract: CurveBase.sol

Function: switchDex

Recommendation: Consider saving the state of the approvals.

Status: Fixed.

2. Function state mutability can be restricted to pure

View functions that don't access the state could be declared as pure to save some gas.

Contracts: CurveBtc.sol, CurveEth.sol, CurveStable.sol, CurveBase.sol

Function: _getCoinsCount, _getWantTokenIndex

Recommendation: Please use the "**pure**" modifier for view functions that don't access the state.

Status: Fixed.

3. State variables could be defined as immutable

State variables that don't change their values and are initialized in the constructor should be defined as immutable to save gas.

Contract: CurveStable.sol

Variable: wantThreepoolIndex

Recommendation: Please use the "**immutable**" modifier for state variables that never change and are initialized in the constructor.

Status: Fixed.

4. State variables could be defined as constant

State variables that don't change their values should be defined as constant to save gas.

Contract: CurveStable.sol

Variable: N_POOL_COINS

Recommendation: Please use the "constant" modifier for state variables that never change.

Status: Fixed.

5. Unused function parameters
- **Contract**: Staking.sol
Function: _beforeTokenTransfer
Parameters: _operator, _amounts, _data
- **Contract**: SingleAssetVault.sol
Function: _authorizeUpgrade
Parameters: implementation
- **Contract**: CurveBase.sol
Function: adjustPosition
Parameters: _debtOutstanding
- **Contract**: CurveBase.sol
Function: prepareMigration
Parameters: _newStrategy
- **Contract**: BaseVault.sol
Function: _beforeTokenTransfer
Parameters: _amount
- **Contract**: BaseStrategy.sol
Function: tendTrigger
Parameters: callCost

Recommendation: While it's an overridden method you may just remove the variable name, leaving the definition in place (ie. *function*

_beforeTokenTransfer(address, address _from, address _to, uint256[] memory _ids, uint256[] memory, bytes memory) internal override).

Status: Fixed.

6. Unused local variable

Contract: CurveStable.sol

Function: _depositLPTokens

Variable: balance

Recommendation: Please remove unused local variable.

Status: Fixed.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found **1** medium and **6** low severity issues.

After the second review security engineers found that an additional interface support check was added to the adding new strategies, flash loan possibility was removed by putting a restriction to the user to transfer any tokens in the same block when tokens are minted and low issues were also addressed. Therefore security engineers didn’t find any security issues with the updated code.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.