HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Pluto Digital, YOP protocol
Date: April 28th, 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	Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type	Vault; Staking
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://plutodigital.com/ https://yop.finance/
Timeline	15.04.2022 - 28.04.2022
Changelog	22.04.2022 - Initial Review 28.04.2022 - 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 contracts.

Scope

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

Initial review scope
Repository:
https://github.com/plutodigital/yop-protocol-evm
Commit:
daea00112015d9f3637c04c6bb577dc24ebd2bf1
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/v2/docs

JS tests: Yes
https://github.com/plutodigital/yop-protocol-evm/tree/v2/test
Contracts:
access/PerVaultGatekeeper.sol
(sha3: 2b01343716987429bd8445e531c3e7b58bb59dea6b1169226afde5d8)
access/AllowListAccessControl.sol
(sha3: d2e25b6a0f45ddf494b9652785daeaa46c1849b3ef434ff03d203a2e)
access/AllowAnyAccessControl.sol
(sha3: fef9b73cbb2a5d1653d10c2fd7146677096ea6cb02265a51ec668991)
access/ERC1155AccessControl.sol
(sha3: f3d115d9a65ae2e14f63fb340f9cb9a8a1c6be457598f9b644e1dcfb)
access/AccessControlManager.sol
(sha3: 6ead966b3296afdd3e89f8c2237af7991421bd3d160d20757ae77148)
access/SanctionsListAccessControl.sol
(sha3: 75e1357e480f0d4dbcbc88019c0ac85221558748fd7a22405f38ae49)
fees/FeeCollection.sol
(sha3: b0de7fbd9b54aea4cf8f3487b37311fb1cb30231ea1bdb5069045198)
security/BaseUpgradeable.sol
(sha3: bf18234547433ca517328cf1856214b3c40d58fcda33155e9a72343e)
security/BasePauseableUpgradeable.sol
(sha3: 8c73a6dbbc569ec6664e37ee1abd89d70eba55960853abe644d85288)
strategies/ConvexBase.sol
(sha3: 5df4f712472c42637158c0ca6e710eee1ad94e5db742def126e363e2)
strategies/curvev2/CurveETHSinglePool.sol
(sha3: 1d2eb5672a425f5b19465b4436a43e8e36458797057f0f709cdf81ff)
strategies/curvev2/CurveMeta.sol
(sha3: 396294a02fe3f33295e166fe65f830ffc9261dd2d5ca868e40fd6bcb)
strategies/curvev2/CurveBaseV2.sol
(sha3: a09caaba564e953a7d7cb038b20c07550cdfeceeb957145ba26d25cb)
strategies/curvev2/CurveERC20SinglePool.sol

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

```
(sha3: d220e20d80cef0a003dfa86880d44138cde65d78b9d98929cdf83519)
strategies/CurveEth.sol
(sha3: 3650169dfa49c62bf6452fbaf1f1c48c3f3b9e978d38f46029e4bcb9)
strategies/convexv2/ConvexCurveMeta.sol
(sha3: 988672fdbaefe17593c1ef87a4e9172141521ace5341563b7a759635)
strategies/convexv2/ConvexETHSinglePool.sol
(sha3: 6d6def576e2618303319e91e0ad5e15b2efd6b1fdb5f4478f417cf21)
strategies/convexv2/ConvexERC20SinglePool.sol
(sha3: 9e11fb65f0a4eeadbdc9c321735fd0f00b5b4190cdcd96f0a609cd11)
strategies/ConvexEth.sol
(sha3: 30cd237105e440fc8cc34b96b5ab02270c6a215f1c75594fa54a52e6)
strategies/BaseStrategy.sol
(sha3: 819c86190559c0782f0e9749b9e2b2e1314fa4ed86c3d8783657af06)
strategies/CurveBtc.sol
(sha3: 6432402838256016047 3a64aecd9b61c222c79a383e9a15cb6f442a8)
strategies/CurveStable.sol
(sha3: 4ddd219bd91eca3702f9153deff940b56e239ba7df32903b8e6399b6)
strategies/ConvexStable.sol
(sha3: 4ca5554f6a1439cef4374f53dcc8f7218bcbf1302c5c790766c73ddf)
strategies/CurveBase.sol
(sha3: 3f60d4b7b0a006ffb8e0fefeb292552352bd71f4b4ec6a370b7d6a95)
strategies/ConvexBtc.sol
(sha3: c66194bb7b7df0d18e5a1acd2d8ab32f3df3c3bb5152109774f78790)
rewards/YOPRewardsV2.sol
(sha3: dfc100274a583ba5d3053d766c6428f57b93e7a0963b427fded5127f)
rewards/YOPRewards.sol
(sha3: 07bf782f111af1dd69a89fd4ebeba4e6905837f38e77988a5d8912ef)
libraries/VaultUtils.sol
(sha3: 9bdc85791e2e8faf5961cea11c1b4cc4af69a841320d34cf180fb763)
libraries/ConvertUtils.sol
(sha3: 4b1ac650acfee98ca891dea149127a63d0f8077f560a69d6c3f4cb5d)
libraries/SwapUtils.sol
(sha3: 306eff3e80a599bdea3b9e302b312ae5acbf471a91a7a5790fb023be)
vaults/CommonHealthCheck.sol
(sha3: 6e8be08fde260beba3ab0abac0a2d2060b99255c68909e108f5bdf28)
vaults/roles/Gatekeeperable.sol
(sha3: 6f6bbb8b6f5e29756a5b0ded1260209e78ed21bc77721e0c5f3324c4)
vaults/roles/Governable.sol
(sha3: d56b07560b1034a4ad16c4289f1246feda49ca94fa78ce9e2b96ef3a)
vaults/roles/Manageable.sol
(sha3: 5872ee10051ce7c5855d3f86ba46e8b378bbba6afe5e7a5e158ec738)
vaults/VaultStrategyDataStore.sol
(sha3: 88a977e9221ed95c769dc950807c3f379e24e02d6085f07d4e35f131)
vaults/SingleAssetVaultBase.sol
(sha3: 3fb5d6a09ac1dc8f1d111ea4b685edfa2aff6d853b754116b35d818d)
vaults/SingleAssetVault.sol
(sha3: c7428005afa1a9c64d386911957f9d7da9201303299fea5acc4cf065)
vaults/VaultDataStorage.sol
(sha3: 8c20ced0030a789a0325ad06afef41c80077f24d2500accf232c6f06)
vaults/SingleAssetVaultV2.sol
(sha3: faf40d4e9269a5198a6dfb9d008e5660d5a2fdc78f9b345a36fde891)
vaults/BaseVault.sol
(sha3: 8c8a64d3868e5ed5c15ed2661707cf77857a9a495d53753c2c7d2002)
vaults/VaultMetaDataStore.sol
(sha3: 14246a1c37bed9d3f5caa67c51a4da0066e93a7158a225197fcb17d2)
staking/StakingV2.sol
```

(sha3: 18cf35c6bf89083d6b07db0d1c122a3adfa29d4348448e7031709b54)
staking/Staking.sol
(sha3: 086343d3a0cad08af35560172945db20d0eff7ae8f91e5f38d018426)
registry/YOPRegistry.sol
(sha3: 3ed517a639af0fb8a318ccf35a529d6bbcf13f7226a0bf1773fad652)
router/YOPRouter.sol
(sha3: 3ebbdb1b65b04fc0ef01a172bf4a1e23e5ee776d75392e536a937d7f)

Second review scope

Repository:
https://github.com/plutodigital/yop-protocol-evm
Commit:
c94e4442a3a60ac79af33c344e41830d81b1dca5
Technical Documentation: Yes

3. Whitepaper:
   https://cdn.yop.finance/wp-content/uploads/2022/01/26140053/YOP_Whitepaper_final.pdf
4. Tech docs
   https://github.com/plutodigital/yop-protocol-evm/tree/v2/docs

JS tests: Yes
https://github.com/plutodigital/yop-protocol-evm/tree/v2/test
Contracts:
access/PerVaultGatekeeper.sol
(sha3: 2b01343716987429bd8445e531c3e7b58bb59dea6b1169226afde5d8)
access/AllowListAccessControl.sol
(sha3: d2e25b6a0f45ddf494b9652785daeaa46c1849b3ef434ff03d203a2e)
access/AllowAnyAccessControl.sol
(sha3: fef9b73cbb2a5d1653d10c2fd7146677096ea6cb02265a51ec668991)
access/ERC1155AccessControl.sol
(sha3: f3d115d9a65ae2e14f63fb340f9cb9a8a1c6be457598f9b644e1dcfb)
access/AccessControlManager.sol
(sha3: 8f2dddeb878f254f4883adc9aedd149cc4c743b72b9bcf8f342ae6dc)
access/SanctionsListAccessControl.sol
(sha3: 75e1357e480f0d4dbcbc88019c0ac85221558748fd7a22405f38ae49)
fees/FeeCollection.sol
(sha3: e24a0204551db6de2baef6959cafc1f5e77170b617802e8e7e0e4de3)
security/BaseUpgradeable.sol
(sha3: bf18234547433ca517328cf1856214b3c40d58fcda33155e9a72343e)
security/BasePauseableUpgradeable.sol
(sha3: 8c73a6dbbc569ec6664e37ee1abd89d70eba55960853abe644d85288)
strategies/ConvexBase.sol
(sha3: 57390e3e94f60f17fef0d291db58cfbcc50649e6734f9d3ec17231d3)
strategies/curvev2/CurveETHSinglePool.sol
(sha3: 00ae4746505a7364ebb9071f8826e70aa2fc1a9b239573281d2a8eac)
strategies/curvev2/CurveMeta.sol
(sha3: 396294a02fe3f33295e166fe65f830ffc9261dd2d5ca868e40fd6bcb)
strategies/curvev2/CurveBaseV2.sol
(sha3: 6e4d2a992fcea86c78abb67f0f3f6845cb1842d039becc6bd35868e7)
strategies/curvev2/CurveERC20SinglePool.sol
(sha3: cfc9289ab86e89880bba10c47c98c67817f39a0cc1f9021c271b858f)
strategies/CurveEth.sol
(sha3: 3650169dfa49c62bf6452fbaf1f1c48c3f3b9e978d38f46029e4bcb9)
strategies/convexv2/ConvexCurveMeta.sol
(sha3: ecffb43cf102983363dfbd2c2ac38211a14597f4c52f320c1996692e)
strategies/convexv2/ConvexETHSinglePool.sol
(sha3: 8d21c5a005cc118089f099c917754ce9f1e5014938ba0f945dd35cb1)

```
strategies/convexv2/ConvexERC20SinglePool.sol
(sha3: 99f88ee871712fdf93e4ec66c7f8b754f53c6c2f3092dfed29a51c1f)
strategies/ConvexEth.sol
(sha3: 30cd237105e440fc8cc34b96b5ab02270c6a215f1c75594fa54a52e6)
strategies/BaseStrategy.sol
(sha3: 7044be9f2c230e3f089ba5523c8e78846a72afe469da56076e3711cf)
strategies/CurveBtc.sol
(sha3: 6432402838256016047 3a64aecd9b61c222c79a383e9a15cb6f442a8)
strategies/CurveStable.sol
(sha3: 4ddd219bd91eca3702f9153deff940b56e239ba7df32903b8e6399b6)
strategies/ConvexStable.sol
(sha3: 4ca5554f6a1439cef4374f53dcc8f7218bcbf1302c5c790766c73ddf)
strategies/CurveBase.sol
(sha3: c47415162fe048410c304f7c9a136e4e050ae84b87ccda07530de1fc)
strategies/ConvexBtc.sol
(sha3: c66194bb7b7df0d18e5a1acd2d8ab32f3df3c3bb5152109774f78790)
rewards/YOPRewardsV2.sol
(sha3: 5143b2da153783eb7fb4871af25d0fbc96158920f536815f48bc2176)
rewards/YOPRewards.sol
(sha3: ae66237b3ac6b30aa6d436c4b22921427cc6e03971bb1fdfad642805)
libraries/VaultUtils.sol
(sha3: 148317c09480631621687afade26853304075bfd593f199c0c2ee002)
libraries/ConvertUtils.sol
(sha3: 4b1ac650acfee98ca891dea149127a63d0f8077f560a69d6c3f4cb5d)
libraries/SwapUtils.sol
(sha3: 306eff3e80a599bdea3b9e302b312ae5acbf471a91a7a5790fb023be)
vaults/CommonHealthCheck.sol
(sha3: 1ad2f4da39896ac86eaa15db4c6998f18f1682d08535c2318aa4ed80)
vaults/roles/Gatekeeperable.sol
(sha3: f9e707452ed3f80f8c6e6ef23e3bac391c838316ffcc223e019c4eac)
vaults/roles/Governable.sol
(sha3: c35e95408788151a0f10552d7446ed408c1424c390084988e908f5f5)
vaults/roles/Manageable.sol
(sha3: 5872ee10051ce7c5855d3f86ba46e8b378bbba6afe5e7a5e158ec738)
vaults/VaultStrategyDataStore.sol
(sha3: fb37897614097200fa5f1f309b8e9998f2e3e09492a4d8b741df62e1)
vaults/SingleAssetVaultBase.sol
(sha3: e190401c156cdbe453520b6b420969ff65ba5f4a1450e0ca9f012f05)
vaults/SingleAssetVault.sol
(sha3: b47e3a0459cda2779da2c9e38d04fff86d01b93282efc4c781d21876)
vaults/VaultDataStorage.sol
(sha3: 91abcf594682ea0fc536f850091cac4e0e283ecc1e834642b22424f9)
vaults/SingleAssetVaultV2.sol
(sha3: 5180823d1ebeca0490d17c2959169eeeac532bc0c9f2fa5667197722)
vaults/BaseVault.sol
(sha3: 18a92a11e9b72c9f992c44978f20e9f5dced9dc6e3dcb5477e2eec13)
vaults/VaultMetaDataStore.sol
(sha3: e3d9245659eb23504557c5add9bd426b7012bbd944349b52b64ed2c5)
staking/StakingV2.sol
(sha3: 18cf35c6bf89083d6b07db0d1c122a3adfa29d4348448e7031709b54)
staking/Staking.sol
(sha3: 086343d3a0cad08af35560172945db20d0eff7ae8f91e5f38d018426)
registry/YOPRegistry.sol
(sha3: 3ed517a639af0fb8a318ccf35a529d6bbcf13f7226a0bf1773fad652)
router/YOPRouter.sol
(sha3: da5d05d9103ff248e393a94ec395a60c2c513d766da33c02386bfc53)
```

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 cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution

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

Executive Summary

The score measurements details can be found in the corresponding section of the methodology.

Documentation quality

The Customer provided superficial functional requirements and technical requirements. The total Documentation Quality score is **10** out of **10**.

Code quality

The total CodeQuality score is **10** out of **10**. Code very good commented. Unit tests were provided. NatSpecs used.

Architecture quality

The architecture quality score is **10** out of **10**. All the logic is split correctly. The architecture is clean and fully readable.

Security score

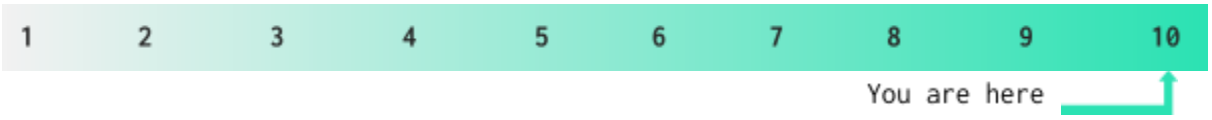
As a result of the audit, security engineers found **2** medium and **5** low severity issues. The security score is **8** out of **10**.

As a result of the second review, security engineers found no new issues. **2** medium and **5** low severity issues from the previous revision were fixed. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **10.0**



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

Checked Items

We have audited provided smart contracts for commonly known and more specific vulnerabilities. Here are some of the items that are considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Passed
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Not Relevant
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be destroyed until it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Uninitialized Storage Pointer	SWC-109	Storage type should be set explicitly if the compiler version is < 0.5.0.	Not Relevant
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Not Relevant
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Not Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Passed
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless it is required.	Passed

Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Passed
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Passed
Signature Unique Id	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	Passed
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes.	Passed
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of unused variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Not Passed
EIP standards violation	EIP	EIP standards should not be violated.	Not Relevant
Assets integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions.	Passed
User Balances manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed
Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Passed
Token Supply manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Passed

Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block gas limit.	Passed
Style guide violation	**Custom**	Style guides and best practices should be followed.	Passed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Repository Consistency	**Custom**	The repository should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed

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

System Overview

YOPprotocol is a mixed-purpose system with the following contracts:

- AccessControlManager – registry for the access control policies that have been added.
- AllowAnyAccessControl – the contract will allow enabling/disabling open access to vaults.
- ERC1155AccessControl – configure and validate user access to a particular vault using ERC1155 NFTs.
- FeeCollection – the contract is used to distribute fees to various participants.
- YOPRegistry – The main on-chain registry to allow querying addresses of YOP components.
- YOPRewards – the contract that will be used to calculate the YOP rewards for vault and staking users based on the emission schedule outlined in YOP tokenomics.
- YOPRewardsV2 – This version of the reward contract will calculate the user's vault rewards using their boosted vault balances (taking staking into account) rather than just their vault balances.
- YOPRouter – the contract allow users to use any tokens against the YOP platform.
- Staking – the contract will stake (lock) YOP tokens for a period of time.
- StakingV2 – added a new stake function that will update the user's boost balance in selected vaults immediately after staking.
- SingleAssetVault – the Vault contract allows to deposit, withdraw and report profits/losses.
- SingleAssetVaultV2 – the contract version that adds support for using "boosted" user balances.
- CurveMeta – the strategy allows to deposit a token into the curve base pool, deposit the LP in the meta pool, and stake it in the gauge.
- ConvexCurveMeta – extends the CurveMeta strategy to use any curve base and meta pool.
- CurveETHSinglePool – a strategy that supports any ETH pool in Curve. This only supports StableSwap (plain) pools. However, all Curve ETH pools are StableSwaps, and only have 2 input tokens, and one of them is ETH.
- CurveERC20SinglePool – a strategy that supports any ERC20 pool in Curve.
- CurveBase – The base implementation for all Curve strategies. All strategies will add liquidity to a Curve pool (a plain or meta pool), and then deposit the LP tokens to the corresponding gauge to earn Curve tokens.
- ConvexBase – the contract provides common functions that will be used by all Convex strategies.

- CurveStable – Implements the strategy using the usdn/3Crv(DAI/USDC/USDT) pool.
- CurveBtc – Implements the strategy using the oBTC/sBTC(renBTC/wBTC/sBTC) pool,
- CurveEth – Implements the strategy using the ETH/stETH swap pool
- ConvexStable – Implements the strategy using Convex. The steps are similar to CurveStable strategy.
- ConvexBtc – Implements the strategy using Convex. The steps are similar to CurveBtc strategy.
- ConvexEth – Implements the strategy using Convex. The steps are similar to CurveEth strategy.

Privileged roles

- The governance role for all contracts has allowed entire access.
- The gatekeeper role has the following access rights:
  - in the Access contracts, it can add and delete control policies, allow/remove vault access, and add/remove NFT mappings;
  - in the Vaults contracts, it can pause, set health check, initiate emergency shutdown, set deposit limit, and set access manager;
  - in the Strategies contracts, it has no access rights;
  - in the Convex contracts, it has no access rights;
- The manager role has the following access rights:
  - in the Access contracts, it has no access rights;
  - in the Vaults contracts, it is allowed to set profit and loss limit ratio, set strategy limits, set checks and disable checks, set maximum total debt ratio, add strategy, update strategies fee and min/max debt harvest, set withdrawal queue, add/remove strategy to withdrawal queue, revoke strategy;
  - in the Strategies contracts, it has no access rights;
  - in the Convex contracts, it has no access rights;
- The strategist role has the following access rights:
  - in the Access contracts, it has no access rights;
  - in the Vaults contracts, it has no access rights;
  - in the Strategies contracts, it has all rights except for migrating and sweeping;
  - in the Convex contracts, it has all rights;
- The keeper role has the following access rights:
  - in the Access contracts, it has no access rights;
  - in the Vaults contracts, it has no access rights;
  - in the Strategies contracts, it could only call tend and harvest functions;
  - in the Convex contracts, it has no access rights;

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

Findings

Critical

No critical severity issues were found.

High

No high severity issues were found.

Medium

1. **Tautology or contradiction.**

   In the function argument, there is a uint16 variable type used. uint means for an unsigned integer. Unsigned means it could never be less than zero. Therefore, the checking for the value to be “>=0” is either excess or a miscoded logic.

   Contract: FeeCollection.sol

   Functions: setVaultCreatorFeeRatio, _setDefaultStrategyFeeRatio, _setDefaultVaultCreatorFeeRatio

   Recommendation: Double check if there are no side effects and remove the tautology.

   Status: Fixed (Revised Commit: c94e444)

2. **Contracts that lock Ether.**

   If someone will ever send Ether directly to the contract address, those Ether would be locked forever.

   Contract: YOPRouter.sol

   Functions: receive

   Recommendation: add a check that the sender is the WETH contract or add a withdrawal function to the contract.

   Status: Fixed (Revised Commit: c94e444)

Low

1. **A public function could be declared external.**

   Public functions that are never called by the contract should be declared **external**.

   Contracts: VaultUtils.sol, BaseVault.sol

   Function: VaultUtils.calculateBoostedVaultBalance, VaultUtils.checkStrategyHealth, VaultUtils.assessFees, VaultUtils.creditAvailable, VaultUtils.debtOutstanding, VaultUtils.expectedReturn, BaseVault.supportsInterface

   Recommendation: Use the **external** attribute for functions never called from the contract.

Status: Fixed (Revised Commit: c94e444)

2. **Unused import.**

   Contracts import the “hardhat/console.sol” but never use it in the code.

   Contracts: CurveBase.sol, ConvexBase.sol, ConvexERC20SinglePool.sol, ConvexETHSinglePool.sol, CurveBaseV2.sol, SingleAssetVault.sol

   Recommendation: remove unused import statements.

   Status: Fixed (Revised Commit: c94e444)

3. **Unused state variables.**

   Contracts declare public state variables that are never initialized in the contract or inheritors.

   Contracts: BaseStrategy.sol, ConvexCurveMeta.sol

   Variable: rewards, convelpTokenxDepoist

   Recommendation: remove unused state variable.

   Status: Fixed (Revised Commit: c94e444)

4. **Reading state in the loop.**

   Calling the `length()` function in the condition statement block of the for-loop makes it to call this function on each iteration.

   Contracts: AccessControlManager.sol

   Function: _hasAccess

   Recommendation: read the value into a local memory variable and use it in the loop.

   Status: Fixed (Revised Commit: c94e444)

5. **Using experimental ABIEncoderV2 pragma.**

   Starting solidity version 0.8.0 the ABI coder v2 is activated by default. Choose the old behavior using `**pragma abicoder v1;**`. The pragma `**pragma experimental ABIEncoderV2;**` is still valid, but it is **deprecated** and has no effect.

   Recommendation: remove the deprecated pragma. To be explicit, use `pragma abicoder v2`; instead.

   Status: Fixed (Revised Commit: c94e444)

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 by the best industry practices at the date of this report, with 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 a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements. 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 cannot guarantee the explicit security of the audited smart contracts.