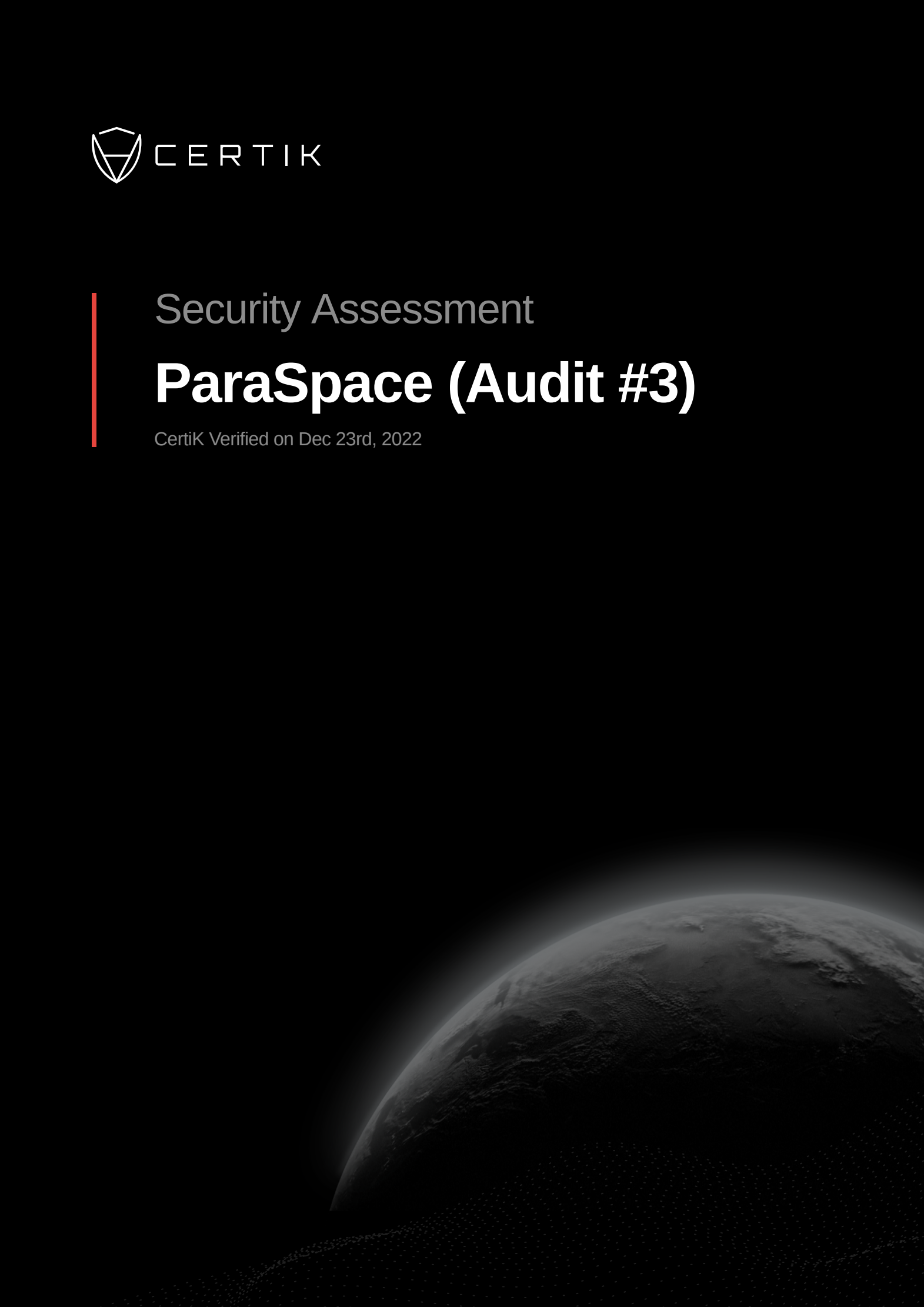
CERTIK

Security Assessment

ParaSpace (Audit #3)

CertiK Verified on Dec 23rd, 2022



CertiK Verified on Dec 23rd, 2022

ParaSpace (Audit #3)

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
NFT	Ethereum	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 12/23/2022	N/A

CODEBASE

https://github.com/para-space/paraspace-core/commit/8026a8addbd01fbd19c66eea59c506dd1ca71467

...View All

COMMITS

8026a8addbd01fbd19c66eea59c506dd1ca71467

...View All

Vulnerability Summary

12	3	0	1	8	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
2	Major	1 Resolved, 1 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
1	Medium	1 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
8	Minor	1 Resolved, 1 Partially Resolved, 6 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
1	Informational	1 Acknowledged	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | PARASPACE (AUDIT #3)

CODEBASE | PARASPACE (AUDIT #3)

Repository

https://github.com/para-space/paraspace-core/commit/8026a8addbd01fbd19c66eea59c506dd1ca71467

Commit

8026a8addbd01fbd19c66eea59c506dd1ca71467

AUDIT SCOPE | PARASPACE (AUDIT #3)

45 files audited ● 21 files with Acknowledged findings ● 24 files without findings

ID	File	SHA256 Checksum
ERC	misc/ERC721OracleWrapper.sol	7f0b939d8632a5d562489d35659b6e1f4ae27084ba36999ae5d21e3cf09df4d8
NFT	misc/NFTFloorOracle.sol	42da955f4b648bfc069fb1e3568d9c0905079bf483ec350043ed74e1844b58e4
PDP	misc/ProtocolDataProvider.sol	044b40cc333e224de24b9d6effb57164985ceadee95c81d72b1fe16c083a232f
UVO	misc/UniswapV3OracleWrapper.sol	e5732332bd7ed36b22cc39a667aff50ee8c4223ca5bf7ee254baee68c92da952
PSO	misc/ParaSpaceOracle.sol	a901e169075cf477420e3afbecc60f43ed198300573b46227e35561addfad748
ACL	protocol/configuration/ACLManager.sol	95cef06ac33289cadab6d5793999801cd026a2ff44da116f9fa03d21417e28a7
PAP	protocol/configuration/PoolAddressesProvider.sol	ea97d8b90dc8b467b28510a6d2ef8b683f14b17a7c29b8234a9589e04a368989
PAR	protocol/configuration/PoolAddressesProviderRegistry.sol	267f4dc860bd1c09577abec389e4689fd09f62c8f8a7f89001c21bdaf7d61ee5
LLB	protocol/libraries/logic/LiquidationLogic.sol	717ec516bda03c68544da274862e3302caa85f7ffdea10a44a0964d0d22c30de
VLB	protocol/libraries/logic/ValidationLogic.sol	a8f4e4f39c8072a40a9dc9bafc31c5f81888720259ec9c6d9a108459871e3b58
PAS	protocol/pool/PoolApeStaking.sol	246a7aa3ebce366dd0c5234cdce0207271a67d170647e607637fbfa3d3b1da00
PCB	protocol/pool/PoolConfigurator.sol	1a9e47b2c68689393263ad4bf8e1ae792da67c12849cd52f46a3c2c570f18a73
PCU	protocol/pool/PoolCore.sol	be62da402977d6218d6592ee4519b8a85477b72f358c6bff05d8b62feb776c7e
PPB	protocol/pool/PoolParameters.sol	c883a89621f490ed93958a6d42f1b2d951a14e27c93d931a7df419a9e4419db3

ID	File	SHA256 Checksum
MIE	protocol/tokenization/base/MintableIncentivizedERC721.sol	31355a0ab3b7be86c83e9f23bb9ae4548ece1d2b6bf05bc2a3407624eedc1040
NTB	protocol/tokenization/NToken.sol	a75e432aa598cbd6fa733c7547ac8114b2b317f7a162faa9ffd9753070972daa
NTA	protocol/tokenization/NTokenApeStaking.sol	f8f9e7180a3284724964f8e57e9613220521783fc51ad924c9d04bf0dc38751d
NTU	protocol/tokenization/NTokenUniswapV3.sol	1805029862deb34bc02808b94b43b51bdc5e6e5549f76ae240d301f4c92726f2
ASL	protocol/tokenization/libraries/ApeStakingLogic.sol	f1e55c1c682ecbd5787d24680618be02ca09dc49009a52fdeb304d875459b3b6
WET	ui/WETHGateway.sol	f594d87684b71801c5d662262ae0b7d211e7a038da3ee6fe32076c49a9c3fce4
WPG	ui/WPunkGateway.sol	95c06c9296e6785a1168a959ca95a86510202cbb58361fcfc3bd0ae82db44d10
AFC	misc/flashclaim/AirdropFlashClaimReceiver.sol	f2ad581f2e781caf9990e9a6226d13cd2ad696f9540a2de8fc3dc0eea4f8e013
UFR	misc/flashclaim/UserFlashclaimRegistry.sol	efb43cd2dcd02a616e02fc76c1ea48cb891e311e332c02574adcf2191d17375f
LRA	misc/marketplaces/LooksRareAdapter.sol	99549b3fc3784af77fc8542db82e7cec27795c1de2cb6ea7057eadf566f2e141
SAB	misc/marketplaces/SeaportAdapter.sol	26bb19b93659dd7579d47e0b37bcfe2fc3f2570b74093ad6c930779d67219a20
XYA	misc/marketplaces/X2Y2Adapter.sol	982f647a0367feba0113defa61dffe7f6c221a03392f2a33d9f5293af051fadd
RCB	protocol/libraries/configuration/ReserveConfiguration.sol	10138e8e6ef8d2fe37ef5d50aedb195f492a8dda53112d002a00f4d45aa96743
UCB	protocol/libraries/configuration/UserConfiguration.sol	86aaf1f476e75a6bc50a08347335c979216a7ff3645045529141227eb083943e
ALB	protocol/libraries/logic/AuctionLogic.sol	3452d7dc84261df0cbda01843516d67d72771fa2cfac1e503c748569d936f1ae
BLB	protocol/libraries/logic/BorrowLogic.sol	71458a74d51f9faa543b14ce7607889df5a7438700a771741e4b082d7b817295

ID	File	SHA256 Checksum
CLB	protocol/libraries/logic/ConfiguratorLogic.sol	8afa98e7f3a5891e6fc21cd39bda33a31a1c5b423a988d0445eb80890b765213
FCL	protocol/libraries/logic/FlashClaimLogic.sol	ed23be494624563200156cb659bbf48aa4da75f040d7ef4318d1b4dd7480acf1
GLB	protocol/libraries/logic/GenericLogic.sol	3892d25301be65ff370e1a7e348d672786fbdf3484fecd69d785eea10a4c6f5f
MLB	protocol/libraries/logic/MarketplaceLogic.sol	2e0d5d843aa9fdb0d2543a066b3f7d630b18c2d9e834b78635fe9fd779af5b06
PLB	protocol/libraries/logic/PoolLogic.sol	1264a82037c8f98762ea99441fcc69236d7fa40ab86dc1d81991c2ac6ed25d27
RLB	protocol/libraries/logic/ReserveLogic.sol	fbc335e45ab38410c14ab1a46e78072955b59f0a4b25839ff2af889eaf7594c2
SLB	protocol/libraries/logic/SupplyLogic.sol	80d3643c8b98d8a82c314fe8132a113782a700f7ea7c86397e66d80c1f94980c
DRA	protocol/pool/DefaultReserveAuctionStrategy.sol	baf80e9a969149111af005ee1576d04c3d09ff79298f169803521ebf418c0081
DRI	protocol/pool/DefaultReserveInterestRateStrategy.sol	2a3ebc29d82d4ae8a291d0fc2a798fec124d13768f13e44c0691840edc2e09ad
PMB	protocol/pool/PoolMarketplace.sol	a707be4268fa3dfd7225c80eb4916f6cccd48f9c33976996de8f28a0e3a53028
PSB	protocol/pool/PoolStorage.sol	614681592943071 87eef941e6d0df3249cb1cd78d84b8bf99daebcbfcc11c88c
NTY	protocol/tokenization/NTokenBAYC.sol	cadb0d4dc9f04e55968bbb6c8e4230882bfa5c48ae82251c482ac6c92418ac78
NTM	protocol/tokenization/NTokenMAYC.sol	a68b86dee27559cf582a5e997406602d767443df20393b45be831422deb784f2
NMB	protocol/tokenization/NTokenMoonBirds.sol	c369a5ce4e4e8dd40bd61a811c1855a9d9a805fd079864ac03e975c6792e96b0
MER	protocol/tokenization/libraries/MintableERC721Logic.sol	03de2cce3ad56abaffac3383c6c397519c6794007d9369d58e34a73996358e04

APPROACH & METHODS | PARASPACE (AUDIT #3)

This report has been prepared for ParaSpace to discover issues and vulnerabilities in the source code of the ParaSpace (Audit #3) project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis 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.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

REVIEW NOTES | PARASPACE (AUDIT #3)

The codebase for this audit is in the below repository:

https://github.com/para-space/paraspace-core

Only the differences from **0924ebdf307957723a8d1195220952b890781d2a** to **8026a8addbd01fbd19c66eea59c506dd1ca71467** were reviewed. The audit scope only includes the delta part between these two commits.

The detailed file list is in the above audit scope section.

FINDINGS | PARASPACE (AUDIT #3)

12	0	2	1	8	1
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for ParaSpace (Audit #3). Through this audit, we have uncovered 12 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
GLOBAL-01	Third Party Dependencies	Logical Issue	Minor	Acknowledged
ASL-01	Withdraw Balance Is Not Properly Calculated	Logical Issue	Minor	Resolved
CON-01	**Centralization Related Risks**	**Centralization / Privilege**	**Major**	**Acknowledged**
CON-02	Unused Return Value	Volatile Code	Minor	Acknowledged
MIE-01	Safe Transfer Not Invoke `_checkOnERC721Received()`	Logical Issue	Minor	Acknowledged
NFT-01	No Access Restriction On Function `removeFeeder()`	Logical Issue	Major	Resolved
NTA-01	Not Withdraw Rewards While Transferring Asset	Logical Issue	Medium	Resolved
NTB-01	No Access Restriction On Function `initialize()`	Logical Issue	Minor	Acknowledged
PDP-01	Not Distinguish Between ERC20 Or ERC721 Token	Logical Issue	Minor	Acknowledged
UVO-01	No Price Validation Between UniswapV3 And ChainLink	Logical Issue	Minor	Partially Resolved

ID	Title	Category	Severity	Status
VLB-01	`auctionStrategyAddress` Is Not Explicitly Checked In `validateEndAuction()`	Volatile Code	Minor	Acknowledged
UVO-02	Potential Price Manipulation	Logical Issue	Informational	Acknowledged

GLOBAL-01 | THIRD PARTY DEPENDENCIES

Category	Severity	Location	Status
Logical Issue	Minor		Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party `UniswapV3`, `OpenSea`, `LooksRare`, `X2Y2`, `MoonBird`, `ChainLink`, `ApeCoinStaking`, `IEACAggregatorProxy`, `IAtomicPriceAggregator` and NFT Oracle protocols. The scope of the audit treats 3rd party entities as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

The `ParaSpace` protocol allows users to borrow assets using NFT as collateral. If NFT prices fluctuate significantly in the third-party markets, the Supplier's health factory may fluctuate as well. This is a potential risk to this protocol and to the Supplier.

Recommendation

We understand that the business logic of `ParaSpace` requires interaction with `UniswapV3`, `OpenSea`, `LooksRare`, `X2Y2`, `MoonBird`, `ChainLink`, `ApeCoinStaking`, `IEACAggregatorProxy`, `IAtomicPriceAggregator` and NFT Oracle protocols, etc. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

ASL-01 | WITHDRAW BALANCE IS NOT PROPERLY CALCULATED

Category	Severity	Location	Status
Logical Issue	Minor	protocol/tokenization/libraries/ApeStakingLogic.sol: 53	Resolved

Description

The withdrawal amount of Ape coins from the staking contract is not properly calculated, this will transfer all Ape coins left in the current contract to the recipient. The token balance should be checked before and after the withdrawal.

```
38      function withdrawBAKC(
39          ApeCoinStaking _apeCoinStaking,
40          uint256 poolId,
41          ApeCoinStaking.PairNftWithAmount[] memory _nftPairs,
42          address _apeRecipient
43      ) external {
44          ApeCoinStaking.PairNftWithAmount[]
45              memory _otherPairs = new ApeCoinStaking.PairNftWithAmount[](0);
46
47          if (poolId == BAYC_POOL_ID) {
48              _apeCoinStaking.withdrawBAKC(_nftPairs, _otherPairs);
49          } else {
50              _apeCoinStaking.withdrawBAKC(_otherPairs, _nftPairs);
51          }
52
53          uint256 balance = _apeCoinStaking.apeCoin().balanceOf(address(this));
54
55          _apeCoinStaking.apeCoin().safeTransfer(_apeRecipient, balance);
56      }
```

Recommendation

We advise the client to check the token balance before and after the withdrawal instead of directly getting the balance of the current contract.

Alleviation

The team heeded our advice and resolved this issue in commit `e1bedae875c4cbb79a55bcb2ed4bca4c5693d985`.

CON-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization / Privilege	Major	misc/ERC721OracleWrapper.sol: 44~46; misc/NFTFloorOracle.sol: 139~141, 148~151, 158~160, 175~177, 183~185, 195~199, 195~199, 221~224; misc/ParaSpaceOracle.sol: 66~70, 74~77; protocol/configuration/ACLManager.sol: 40~43; protocol/configuration/PoolAddressesProvider.sol: 56~59, 70~73, 81~84, 105~109, 121~125, 142~145, 158, 170, 182~185, 224~227, 235, 242~248; protocol/configuration/PoolAddressesProviderRegistry.sol: 47~50, 72~75; protocol/pool/PoolConfigurator.sol: 92, 98~100, 105~107, 112~114, 119~122, 131~137, 184~187, 198~201, 224~227, 242~245, 263~266, 277~280, 291~294, 309~312, 327~330, 356~359; protocol/pool/PoolParameters.sol: 110~116, 138~143, 151~154, 166~169, 181~184, 206~210; protocol/tokenization/base/MintableIncentivizedERC721.sol: 131~133, 142; ui/WETHGateway.sol: 195~199, 210~212; ui/WPunkGateway.sol: 202~206, 217~219	Acknowledged

Description

In the contract `ACLManager`, the role `DEFAULT_ADMIN_ROLE` has authority over the following functions:

- function setRoleAdmin(), to set the role as admin of a specific role.

Any compromise to the `DEFAULT_ADMIN_ROLE` account may allow a hacker to take advantage of this authority.

In the contract `PoolAddressesProvider`, the role `Owner` has authority over the following functions:

- function setMarketId(), to associate an id with a specific PoolAddressesProvider.
- function setAddress(), to set an address for an id replacing the address saved in the addresses map.
- function setAddressAsProxy(), to update the implementation of a proxy registered with a certain `id`. If there is no proxy registered, it will instantiate one and set as implementation the `newImplementationAddress`.
- function setPoolImpl(), to update the implementation of the Pool or creates a proxy setting for the new `pool` implementation when the function is called for the first time.
- function setPriceOracle(), to update the address of the price oracle.
- function setACLManager(), to update the address of the ACL manager.
- function setACLAdmin(), to update the address of the ACL admin.
- function setPriceOracleSentinel(), to update the address of the price oracle sentinel.

- function setPoolDataProvider(), to update the address of the data provider.
- function setWETH(), to update the address of the WETH.
- function setMarketplace(), to update the info of the marketplace.

Any compromise to the `Owner` account may allow a hacker to take advantage of this authority.

In the contract `PoolAddressesProviderRegistry`, the role `Owner` has authority over the following functions:

- function registerAddressesProvider(), to register an addresses provider.
- function unregisterAddressesProvider(), to remove an addresses provider from the list of registered addresses providers.

Any compromise to the `Owner` account may allow a hacker to take advantage of this authority.

In the contract `PoolConfigurator`, the role `onlyPoolAdmin` has authority over the following functions:

- function dropReserve(), to drop a reserve entirely.
- function updatePToken(), to update the PToken implementation for the reserve.
- function updateStableDebtToken(), to update the stable debt token implementation for the reserve.
- function updateVariableDebtToken(), to update the variable debt token implementation for the asset.
- function setReserveActive(), to activate or deactivate a reserve.

Any compromise to the `onlyPoolAdmin` account may allow a hacker to take advantage of this authority.

In the contract `PoolConfigurator`, the role `onlyRiskOrPoolAdmins` has authority over the following functions:

- function setReserveBorrowing(), to configure borrowing on a reserve.
- function configureReserveAsCollateral(), to configure the reserve collateralization parameters.
- function configureReserveAsAuctionCollateral(), to configure the reserve collateralization parameters.
- function setReserveStableRateBorrowing(), to enable or disable stable rate borrowing on a reserve.
- function setReserveFreeze(), to freeze or unfreeze a reserve. A frozen reserve doesn't allow any new supply, borrow, or rate swap but allows repayments, liquidations, rate rebalances, and withdrawals.
- function setReserveFactor(), to update the reserve factor of a reserve.
- function setSiloedBorrowing(), to set siloed borrowing for an asset.
- function setBorrowCap(), to update the borrowing cap of a reserve.
- function setSupplyCap(), to update the supply cap of a reserve.
- function setLiquidationProtocolFee(), to update the liquidation protocol fee of the reserve.
- function setReserveInterestRateStrategyAddress(), to set the interest rate strategy of a reserve.

Any compromise to the `onlyRiskOrPoolAdmins` account may allow a hacker to take advantage of this authority.

In the contract `PoolConfigurator`, the role `onlyEmergencyOrPoolAdmin` has authority over the following functions:

- function setReservePause(), to pause a reserve. A paused reserve does not allow any interaction (supply, borrow, repay,
    - swap interest rate, liquidate, NToken/PToken transfers).

Any compromise to the `onlyEmergencyOrPoolAdmin` account may allow a hacker to take advantage of this authority.

In the contract `PoolConfigurator`, the role `onlyEmergencyAdmin` has authority over the following functions:

- function setPoolPause(), to pause or unpause all the protocol reserves. In the paused state all the protocol interactions are suspended.

Any compromise to the `onlyEmergencyAdmin` account may allow a hacker to take advantage of this authority.

In the contract `PoolConfigurator`, the role `onlyAssetListingOrPoolAdmins` has authority over the following functions:

- function initReserves(), to initialize multiple reserves.

Any compromise to the `onlyAssetListingOrPoolAdmins` account may allow a hacker to take advantage of this authority.

In the contract `PoolParameters`, the role `onlyPoolConfigurator` has authority over the following functions:

- function initReserve(), to initialize a reserve, activate it, assign an `NToken`/`PToken` and debt tokens and an interest rate strategy.
- function dropReserve(), to drop a reserve.
- function setReserveInterestRateStrategyAddress(), to update the address of the interest rate strategy contract.
- function setReserveAuctionStrategyAddress(), to update the address of the auction strategy contract.
- function setConfiguration(), to set the configuration bitmap of the reserve as a whole.
- function setAuctionRecoveryHealthFactor(), to set the auction recovery health factor.

Any compromise to the `onlyPoolConfigurator` account may allow a hacker to take advantage of this authority.

In the contract `MintableIncentivizedERC721`, the role `onlyPoolAdmin` has authority over the following functions:

- function setIncentivesController(), to set a new Incentives Controller.
- function setBalanceLimit(), to set a new Balance Limit.

Any compromise to the `onlyPoolAdmin` account may allow a hacker to take advantage of this authority.

Any compromise to the `onlyPoolConfigurator` account may allow a hacker to take advantage of this authority.

In the contract `WETHGateway`, the role `Owner` has authority over the following functions:

- function emergencyTokenTransfer(), to transfer ERC20 from the utility contract.
- function emergencyEtherTransfer(), to transfer native Ether from the utility contract.

Any compromise to the `Owner` account may allow a hacker to take advantage of this authority.

In the contract `WPunkGateway`, the role `Owner` has authority over the following functions:

- function emergencyTokenTransfer(), to transfer ERC721 from the utility contract.
- function emergencyEtherTransfer(), to transfer native Punk from the utility contract.

Any compromise to the `Owner` account may allow a hacker to take advantage of this authority.

In the contract `ERC721OracleWrapper`, the role `onlyAssetListingOrPoolAdmins` has authority over the following functions:

- function setOracle(), to set the Oracle contract address.

Any compromise to the `onlyAssetListingOrPoolAdmins` account may allow a hacker to take advantage of this authority.

In the contract `NFTFloorOracle`, the role `DEFAULT_ADMIN_ROLE` has authority over the following functions:

- function addAssets(), to add assets.
- function removeAsset(), to remove an asset.
- function addFeeders(), to add feeders.
- function setConfig(), to update oracle configs.
- function setPause(), to pause an asset.
- function setPrice(), to set a new price on PriceInformation and update the internal Median cumulative price.

Any compromise to the `DEFAULT_ADMIN_ROLE` account may allow a hacker to take advantage of this authority.

In the contract `NFTFloorOracle`, the role `UPDATER_ROLE` has authority over the following functions:

- function setPrice(), to set a new price on PriceInformation and update the internal Median cumulative price.
- function setMultiplePrices(), to set a new price on PriceInformation and update the internal Median cumulative price.

Any compromise to the `UPDATE_ROLE` account may allow a hacker to take advantage of this authority.

In the contract `ParaSpaceOracle`, the role `onlyAssetListingOrPoolAdmins` has authority over the following functions:

- function setAssetSources(), to set or replace price sources of assets.
- function setFallbackOracle(), to set the fallback oracle address.

Any compromise to the `onlyAssetListingOrPoolAdmins` account may allow a hacker to take advantage of this authority.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend

centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The team acknowledged this issue and they stated that they will use timelock to control all the owner functions.

CON-02 | UNUSED RETURN VALUE

Category	Severity	Location	Status
Volatile Code	Minor	protocol/libraries/logic/LiquidationLogic.sol: 472~476; protocol/pool/PoolCore.sol: 436~454, 467~485; protocol/tokenization/NTokenApeStaking.sol: 45~46; protocol/tokenization/NTokenUniswapV3.sol: 72~74; protocol/tokenization/base/MintableIncentivizedERC721.sol: 494~500; ui/WETHGateway.sol: 42, 82, 113, 173; ui/WPunkGateway.sol: 112	Acknowledged

Description

The return value of an external call is not stored in a local or state variable.

Recommendation

We recommend checking or using the return values of all external function calls.

Alleviation

The team acknowledged this issue and they will fix it in their own timeframe.

MIE-01 | SAFE TRANSFER NOT INVOKE `_checkOnERC721Received()`

Category	Severity	Location	Status
Logical Issue	Minor	protocol/tokenization/base/MintableIncentivizedERC721.sol: 320~325	Acknowledged

Description

The function `_safeTransfer()` does not invoke {IERC721Receiver-onERC721Received} on a target address, which is against the EIP-721 standard.

```
320     function _safeTransfer(
321         address from,
322         address to,
323         uint256 tokenId,
324         bytes memory
325     ) internal virtual {
326         _transfer(from, to, tokenId);
327     }
```

Recommendation

We recommend reviewing the logic again and ensure it is intended.

Alleviation

The team acknowledged this issue and they stated that this is by design.

NFT-01 | NO ACCESS RESTRICTION ON FUNCTION `removeFeeder()`

Category	Severity	Location	Status
Logical Issue	Major	misc/NFTFloorOracle.sol: 167~169	Resolved

Description

The function `removeFeeder()` in the aforementioned line can be called by anyone as it has no access restriction. This enables anyone to call this function to remove the price feeder and revoke the `UPDATE_ROLE` of the feeder.

Attack Scenario

Anyone is free to remove any Oracle price feeder.

Proof of concept

PoC exploit code is not necessary in this case. The finding is straightforward and about access restriction on function.

Recommendation

We advise the client to allow only `DEFAULT_ADMIN_ROLE` to call the `removeFeeder()` function.

Alleviation

The team heeded our advice and resolved this issue in commit `2646065009697188d1c6a70b9f7e66e08f6231f6`.

NTA-01 NOT WITHDRAW REWARDS WHILE TRANSFERRING ASSET

Category	Severity	Location	Status
Logical Issue	Medium	protocol/tokenization/NTokenApeStaking.sol: 20	Resolved

Description

The functions `transfer()` and `transferFrom()` are not overridden to withdraw all staked and pending rewards before transferring the asset. So the original owner of the asset will potentially lose some staking and reward tokens.

Proof of concept

1. User A supplies the BAYC/MAYC and gets the BAYC/MAYC NToken, then stakes the APE with their BAYC/MAYC NToken.
2. User A transfers the BAYC/MAYC NToken to User B. User A will lose the staked APE coins and rewards.

Recommendation

We advise the client to consider calling the function `ApeStakingLogic.executeUnstakePositionAndRepay()` upon transferring the asset.

Alleviation

The team heeded our advice and resolved this issue in commit `7fc60926d25c2caa24d90e25a4406544aec64c6e`.

NTB-01 | NO ACCESS RESTRICTION ON FUNCTION `initialize()`

Category	Severity	Location	Status
Logical Issue	Minor	protocol/tokenization/NToken.sol: 60	Acknowledged

Description

The below `require` statement is invalid because the function caller can input the same address as the contract **POOL** to bypass the validation. Hence anyone can call the function `initialize()` to initialize the contract **NToken**.

```
60   require(initializingPool == POOL, Errors.POOL_ADDRESSES_DO_NOT_MATCH);
```

Recommendation

We recommend reviewing the logic again and adding a reasonable restriction on the function.

Alleviation

The team acknowledged this issue and they will leave it as it is for now. They stated the following:

"They will call initialize only when they upgrade and upgrade is only callable by one contract."

PDP-01 | NOT DISTINGUISH BETWEEN ERC20 OR ERC721 TOKEN

Category	Severity	Location	Status
Logical Issue	Minor	misc/ProtocolDataProvider.sol: 196	Acknowledged

Description

The function `getReserveData()` is used to get the reserve data, however, there is no logical distinction between ERC20 and ERC721 assets.

```
189  {
190          DataTypes.ReserveData memory reserve = IPool(
191              ADDRESSES_PROVIDER.getPool()
192          ).getReserveData(asset);
193
194          return (
195              reserve.accruedToTreasury,
196              IERC20Detailed(reserve.xTokenAddress).totalSupply(),
197              IERC20Detailed(reserve.variableDebtTokenAddress).totalSupply(),
198              reserve.currentLiquidityRate,
199              reserve.currentVariableBorrowRate,
200              reserve.liquidityIndex,
201              reserve.variableBorrowIndex,
202              reserve.lastUpdateTimestamp
203          );
204      }
```

Recommendation

We recommend adopting the different logical implementations for ERC20 and ERC721 tokens.

Alleviation

The team acknowledged this issue and they stated that there is no need to distinguish.

UVO-01 NO PRICE VALIDATION BETWEEN UNISWAPV3 AND CHAINLINK

Category	Severity	Location	Status
Logical Issue	Minor	misc/UniswapV3OracleWrapper.sol: 144~150	Partially Resolved

Description

In the function `getTokenPrice()`, the price from the ChainLink is used to calculate the amount0 and amount1 of Token0 and Token1. If the current price of ChainLink is out of the range of the position's lower and upper price, the amount0 or amount1 may be zero, which affects the price of the LP token finally.

```
137  function getTokenPrice(uint256 tokenId) public view returns (uint256) {
138          UinswapV3PositionData memory positionData = getOnchainPositionData(
139              tokenId
140          );
141
142          PairOracleData memory oracleData =
_getOracleData(positionData);//ChainLink
143
144          (uint256 liquidityAmount0, uint256 liquidityAmount1) = LiquidityAmounts
145              .getAmountsForLiquidity(
146                  oracleData.sqrtPriceX96,
147                  TickMath.getSqrtRatioAtTick(positionData.tickLower),
148                  TickMath.getSqrtRatioAtTick(positionData.tickUpper),
149                  positionData.liquidity
150              );
151
152          (
153              uint256 feeAmount0,
154              uint256 feeAmount1
155          ) = getLpFeeAmountFromPositionData(positionData);
156
157          return
158              (((liquidityAmount0 + feeAmount0) * oracleData.token0Price) /
159                  10**oracleData.token0Decimal) +
160              (((liquidityAmount1 + feeAmount1) * oracleData.token1Price) /
161                  10**oracleData.token1Decimal);
162      }
```

Proof of concept

- Case 1: Liquidate Asset

1. A liquidator can call the function `PoolCore.liquidateERC721()` to liquidate a borrower's non-healthy asset with Health Factor below 1.
2. Before the liquidation, the function `calculateUserAccountData()` calculates the total `xToken` balance of the user in the based currency used by the price oracle function `IPriceOracleGetter(oracle).getTokenPrice()` .
3. If the price obtained from ChainLink is at the left of `positionData.tickLower` , the number of token1 obtained will be 0. The price of the LP obtained according to the L158-L161 calculation logic may be lower than the price of the LP itself, which will eventually reduce the health factor of the asset.
4. In this case, the liquidator may successfully liquidate the asset that should be healthy.

- Case 2: Borrow Asset

1. A user who provided collateral can call the function `PoolCore.borrow()` to borrow the ERC20 asset.
2. Before the borrowing, the function `calculateUserAccountData()` calculates the total `xToken` balance of the user in the based currency used by the price oracle function `IPriceOracleGetter(oracle).getTokenPrice()` .
3. If the price obtained from ChainLink is within the range of `positionData.tickLower` and `positionData.tickUpper` , but `currentTick` is exactly equal to `positionData.tickLower` (the number of token1 is 0), according to the calculation logic of L158-L161 the LP price may be higher than the actual price of LP, which will eventually increase the health factor of the asset.
4. In this case, the user may successfully borrow more ERC20 assets than the collateral.

Recommendation

We recommend adding validations between ChainLink's current price and the UniswapV3 position data of the specific LP token, to ensure that ChainLink's price is within LP's price range.

Alleviation

The team acknowledged this issue and confirmed that it would not have a significant impact. They explained that when the price is out of range, the amounts will shift completely to one of the tokens. However, this is a natural price movement and the token will be priced fairly since we can take that LP position and decompose it and get the expected value out of it. Since UniswapV3 on-chain position data can be manipulated, they cannot rely on it. Instead, they use low LTV and LT to minimize the potential impact of this issue. The current implementation aligns with the original project design.

Doc

VLB-01 `auctionStrategyAddress` IS NOT EXPLICITLY CHECKED IN `validateEndAuction()`

Category	Severity	Location	Status
Volatile Code	Minor	protocol/libraries/logic/ValidationLogic.sol: 826	Acknowledged

Description

`validateEndAuction()` requires `isAuctioned(tokenId)`, however, doesn't require

```
require(collateralReserve.auctionStrategyAddress != address(0),
Errors.AUCTION_NOT_ENABLED);
```

Theoretically, it can be turned off via a call to `PoolConfigurator.setReserveAuctionStrategyAddress()`.

Recommendation

We recommend explicitly checking that auction is enabled for a better error handling.

Alleviation

The team acknowledged this issue and they stated the following:

"This is a centralization risk, but they will use time lock to control the centralized functions."

UVO-02 | POTENTIAL PRICE MANIPULATION

Category	Severity	Location	Status
Logical Issue	Informational	misc/UniswapV3OracleWrapper.sol: 74	Acknowledged

Description

Flash loans are a way to borrow large amounts of money for a certain fee. The requirement is that the loans need to be returned within the same transaction in a block. If not, the transaction will be reverted.

An attacker can use the borrowed money as the initial funds for an exploit to enlarge the profit and/or manipulate the token price in the decentralized exchanges.

We find that the function `getOnchainPositionData()` relies on price calculations based on the current sqrt price, meaning that they would be susceptible to flash-loan attacks by manipulating the price of given pairs to the attacker's benefit.

```
71   (uint160 currentPrice, int24 currentTick, , , , , ) = pool.slot0();
```

The result of this function could impact the results of the below functions:

- getLiquidityAmount()
- getLiquidityAmountFromPositionData()
- getLpFeeAmount()
- getLpFeeAmountFromPositionData()

Two functions among them are used in the `UiPoolDataProvider.sol`.

- getLiquidityAmountFromPositionData()
- getLpFeeAmountFromPositionData()

Although the current usage of the abovementioned functions is only on UI, we would like to remind the client to be cautious about the usage of these functions.

Recommendation

We recommend using a time-weighted average price and adding validation to ensure that prices do not fluctuate dramatically over time. For example, use the Observation data to limit the price fluctuation range.

Alleviation

The team acknowledged this issue and they stated the following:

"They only use on-chain position data for UI."

OPTIMIZATIONS | PARASPACE (AUDIT #3)

ID	Title	Category	Severity	Status
CON-03	Unchecked Value Of ERC-20 `transfer()` / `transferFrom()` Call	Volatile Code	Optimization	Acknowledged
NFT-02	Redundant Modifier `onlyWhenAssetExisted` On Function `removeAsset()`	Logical Issue	Optimization	Resolved
PAS-01	Redundant Codes	Coding Style	Optimization	Acknowledged

CON-03 UNCHECKED VALUE OF ERC-20 `transfer()` / `transferFrom()` CALL

Category	Severity	Location	Status
Volatile Code	Optimization	protocol/tokenization/libraries/ApeStakingLogic.sol: 67~71, 233~237; ui/WETHGateway.sol: 81, 172	Acknowledged

Description

The linked `transfer()` / `transferFrom()` invocations do not check the return value of the function call, which should yield `true` in the case of proper ERC-20 implementation.

Recommendation

Since some ERC-20 tokens return no values and others return a `bool` value, they should be handled with care. We advise using the OpenZeppelin's `SafeERC20.sol` implementation to interact with the `transfer()` and `transferFrom()` functions of external ERC-20 tokens. The OpenZeppelin implementation checks for the existence of a return value and reverts if `false` is returned, making it compatible with all ERC-20 token implementations.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

NFT-02 REDUNDANT MODIFIER `onlyWhenAssetExisted` ON FUNCTION `removeAsset()`

Category	Severity	Location	Status
Logical Issue	Optimization	misc/NFTFloorOracle.sol: 151	Resolved

Description

The modifier `onlyWhenAssetExisted` is redundant on the function `removeAsset()`, because it is declared on the internal function `_removeAsset()`.

Recommendation

We recommend removing the redundant modifier.

Alleviation

The team heeded our advice and resolved this issue in commit `4d4d6eaf799db2eed8a2ee0d679a231a7da52c80`.

PAS-01 | REDUNDANT CODES

Category	Severity	Location	Status
Coding Style	Optimization	protocol/pool/PoolApeStaking.sol: 71, 77	Acknowledged

Description

The linked codes do not affect the functionality of the codebase and appear to be either remnant of test code or older functionality.

Recommendation

We recommend removing the redundant code to better prepare the code for production environments.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

APPENDIX | PARASPACE (AUDIT #3)

Finding Categories

Categories	Description
Centralization / Privilege	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE

FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.