CERTIK

Security Assessment

Vinci Protocol - Audit

Aug 26th, 2022

Table of Contents

Summary

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

The auditing process pays special attention to the following considerations:

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

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:

- 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.

Overview

Project Summary

Project Name	Vinci Protocol - Audit
Platform	EVM Compatible
Language	Solidity
Codebase	Github Repo: https://github.com/VinciProtocol/vinci-protocol/
Commit	41ca596f39ba104033c6666f18ca53710554bfcd

Audit Summary

Delivery Date	Aug 26, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	4	0	0	1	3	0	0
Medium	5	0	0	0	0	0	5
Minor	9	0	0	7	0	0	2
Informational	6	0	0	5	0	0	1
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
GPS	projects/vinci-protocol/contracts/dependencies/gnosis/contracts/GPv2SafeERC20.sol	1c47987d702997ff6ffad8e5e0ace5b275842aa8331c97dfa83f2c152bfb234c
ACK	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/AccessControl.sol	4dc760cd150148d1e2dd8818855a50bc7438a6c96a8f88b0b29a6734b630ef90
ACP	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/Address.sol	1563ee0324bcdabdbb92e07b8786563ab7554bf63bced1071367165bef1a9e7e
CCK	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/Context.sol	5a795fa7978a48309df6b77252bb2a38323ec76d8e360761f2d4dbc6df88735e
ERK	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/ERC1155.sol	e4255d3e0b3b458a697b6bff12a7be4405cbf20e1d57ae0bf4ef2b5bd3b222ef
ERP	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/ERC165.sol	e1d8e160d9baa2b583b3dc071d012a70df2e54f848cf31d462ddae7c30312a92
ECC	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/ERC20.sol	3c230db5778846d4cecb78b2431d099a64fee6108375342611562518e062f41c
ECK	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/ERC721.sol	314b7aa9ab3ae86e1d5c477fd3f5d354cec3b9a76f0bc9a051654be0c57280bb
ERE	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/ERC721Enumerable.sol	9124e39b48a6242cd82b29eff954506d8367cb3608c7f9f802117a74b39feefd
EMC	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/EnumerableMap.sol	155c4391ce820278b20b4db4601a5b415d2e8aba507b08b1bac30fa3497c9ffc
ESC	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/EnumerableSet.sol	6328f5661ddf0785e5211b16d23ddcdb9cb9f39d39d8369da9020533d0cef89d
IAC	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/IAccessControl.sol	6fae0696b6b6defad9e1fba0543f48441d620e7b13a38eb35e2c80673ec8eb39
IRK	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/IERC1155.sol	e7db5232210c485d7251a3cccd5db95c0c0e7f3b92b712ed6a7e1f1d2fc47d9c
IEM	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/IERC1155MetadataURI.sol	0e3a99b7fdc8f9bb3b763c895efd266a63ac12d246c45da85eae5a0d925fcd8b

ID	File	SHA256 Checksum
IRR	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/IERC1155Receiver.sol	5dfb6b5f2815dcd5c24eb7ffa0281869ba278a4b13593f3c50e333e6f78100ae
IRP	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/IERC165.sol	ff63231eadeb8a34602d0da93a87c2bacf2c11c739b6684901088457770d51e1
ICC	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/IERC20.sol	8ec7939295a1a7ef8422d4d3896684c0f0fe5006954a7a8ec21823a3d927d22d
IRM	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/IERC20Metadata.sol	5d0769d3fd259dd033417036308c1758be8d7bfbff016ccee663c8548a7f236c
ICK	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/IERC721.sol	50e0e81728ea2aebd2a4e9d0c636c176db2794476f9fe38401f9fb1a0c17be82
IEE	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/IERC721Enumerable.sol	4e2c92be230e11c9ae4e9a294af2ed979a83816ccc774df873c1d2ed26acc1c7
ICM	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/IERC721Metadata.sol	d9ba79674c3134cec28ff6498b459074e586b1e116383d2e3d717b0e9e16c279
ICR	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/IERC721Receiver.sol	80a394bdc47055d9f5204c6fd7af6fa3616e68cf57014ecb5251030da7411c81
OCK	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/Ownable.sol	4303fc3013ac3c36602d6b6d68e4c2399d8d486b050d8b5e8f6f298b687d0945
PCK	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/Pausable.sol	0f2ac1b63e0a54fbd67c29becd0c9e5488100bcb6ed463df1e4ab668449e15aa
RGC	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/ReentrancyGuard.sol	2298aaeeab5871848c81bad7b7dd5836fee0093cf679fc969ee30088f5971070
SCC	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/SafeCast.sol	ada88576131902db866504c4ddcde84e74a4dca0f92a8538607de3b9c4fd8735
SMC	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/SafeMath.sol	bb15a22cb1d30a17ba96a9c570ed0a9ff58ed6e7ab9f0b5af0958bae19b0c522
SSC	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/StorageSlot.sol	17fdfe1fc6739b3e600bc65b4fe1a23321475da8b94fb88680ef36ed00cf386b
SCK	projects/vinci-protocol/contracts/dependencies/openzeppelin/contracts/Strings.sol	7502d0be9762cdb687e70555384af36a59f1b5f54b39cc8d33805d36cc4a3b64

ID	File	SHA256 Checksum
AUP	projects/vinci-protocol/contracts/dependencies/openzeppelin/upgradeability/AdminUpgradeabilityProxy.sol	710c66a5cb57444158d3ef4dc2fec67c47dc1f5973780e7244df9ceb91615b50
BAU	projects/vinci-protocol/contracts/dependencies/openzeppelin/upgradeability/BaseAdminUpgradeabilityProxy.sol	47fb8954d416c7e5819fdf0bd74e2738f15bdfd4985830a490e91daf194c0c89
BUP	projects/vinci-protocol/contracts/dependencies/openzeppelin/upgradeability/BaseUpgradeabilityProxy.sol	4c8269e34beec5329fe5cef686c801b82d9de7b6e8068abd6df8737c68b2bdc5
ECP	projects/vinci-protocol/contracts/dependencies/openzeppelin/upgradeability/ERC1967Proxy.sol	812090e9e79cbddd271214751a0e00f6bbea6a21b6aa26a2b8f9eb57f059a318
ERU	projects/vinci-protocol/contracts/dependencies/openzeppelin/upgradeability/ERC1967Upgrade.sol	8d10ba740c2bc93aedbab1484fd28439ae2a80bcbfeb750e6581c7bb370b8c4f
IBC	projects/vinci-protocol/contracts/dependencies/openzeppelin/upgradeability/IBeacon.sol	b51943a1aa789fb430847d777f1ce248f036aa5bda26a54e60e65dbfa815a1d7
ICP	projects/vinci-protocol/contracts/dependencies/openzeppelin/upgradeability/Initializable.sol	7328d8ea8686ba24cf3d903dea59f06c1ec34c3887c1a6fd27cbc0890013fa20
IAU	projects/vinci-protocol/contracts/dependencies/openzeppelin/upgradeability/InitializableAdminUpgradeabilityProxy.sol	758e33bfbc7601d2062be6e9311f8867f81343d653b76ee7ddb2330fe9237493
IUK	projects/vinci-protocol/contracts/dependencies/openzeppelin/upgradeability/InitializableUpgradeabilityProxy.sol	e747fea2616a13eb0703b622c6d93da9df1f69dc501538dd410fa7313699f87c
PCP	projects/vinci-protocol/contracts/dependencies/openzeppelin/upgradeability/Proxy.sol	95d2bf7e269eac971f221d92bf4b1adf8f8e9b4d895e8bb2775902eb4b69a6fa
PAC	projects/vinci-protocol/contracts/dependencies/openzeppelin/upgradeability/ProxyAdmin.sol	7b7075cf34926836ae58cb02089c74f3ee3ad37671c5264fece830d55e36678f
TUP	projects/vinci-protocol/contracts/dependencies/openzeppelin/upgradeability/TransparentUpgradeableProxy.sol	f0d778b028a49bacd374a7035629b2f24ac1fcb93d6337446fcb8dd1fc6a1184
UPC	projects/vinci-protocol/contracts/dependencies/openzeppelin/upgradeability/UpgradeabilityProxy.sol	f4f6be34c37159346e4c7a8eb641915d101464b42e30d226c9354473b107292f
WEC	projects/vinci-protocol/contracts/dependencies/weth/WETH9.sol	337e55e53442869dbb6d584fc93068752b5fec410f2be656c0136dd159447367
NFT	projects/vinci-protocol/contracts/flashloan/base/NFTFlashLoanReceiverBase.sol	a2929a0d37a8cd372873ccb0a01d255f8018f4a6f466a936383e383efd9181ef

ID	File	SHA256 Checksum
INL	projects/vinci-protocol/contracts/flashloan/interfaces/INFTFlashLoanReceiver.sol	6b87bebb5f653a01258267d2ef152263cf957c689be43470a50b2977f4d80660
IAI	projects/vinci-protocol/contracts/interfaces/IAaveIncentivesController.sol	057d37c416764e4869f4c98f5a0d98cff53e3a1b87bf8de3f01725b5226f479c
IAO	projects/vinci-protocol/contracts/interfaces/IAaveOracle.sol	516ea04456d6a24636b55fde305f87fd79127eb565c424a5a465c37c9a544fd6
ICA	projects/vinci-protocol/contracts/interfaces/IChainlinkAggregator.sol	1ebb5b625db83e1be7926f3e947da0d4b3d53badfb32390318a73037e9b2da15
ICD	projects/vinci-protocol/contracts/interfaces/ICreditDelegationToken.sol	2c5f205477d287285d28afea390b4e73da7603b125cce5c31ff45da633362dc2
IDT	projects/vinci-protocol/contracts/interfaces/IDelegationToken.sol	b3642bd07fd7a8e7d91b4b10c483a9ee59b78ca3b8403329707e4814801a7548
IER	projects/vinci-protocol/contracts/interfaces/IERC165Upgradeable.sol	acf44506f74215316f307757807a066af76a830c33bfd6d3bd405f0aa7210f43
IEC	projects/vinci-protocol/contracts/interfaces/IERC20WithPermit.sol	bb2417a92ca7e5fe49213ff05360ba5893b9ea074b646ae0a9b8e3f1c59f7a64
IEW	projects/vinci-protocol/contracts/interfaces/IERC721WithStat.sol	5a709ac2afb3ca24ebc615be74d6ba092792917ef06b023b9d9ad9cac8743c2a
IEK	projects/vinci-protocol/contracts/interfaces/IERC721Wrapper.sol	63deff87fc3798c55154fc4af9c078b44f0a34b99b5f474a4eb3b8c1b6f541d2
IEA	projects/vinci-protocol/contracts/interfaces/IExchangeAdapter.sol	5e029bfe5528940ee091607148985536f2740fb0b2ed30a7a36d6c4262e4b135
IID	projects/vinci-protocol/contracts/interfaces/IInitializableDebtToken.sol	687cd7ee44c044882824ac48fefb014a2a659fbb4c9fbe893cacd5d9b4c4d0ca
IIN	projects/vinci-protocol/contracts/interfaces/IInitializableNToken.sol	8243139300fca0b7a737873c419d1fb459b9e650bb42f521ada2595de16b408c
IIV	projects/vinci-protocol/contracts/interfaces/IInitializableVToken.sol	712ee845c98c4ff8d94e0ad9c07a84d35d4e7b136863e85e73f3a0c32ba5fcb9
ILP	projects/vinci-protocol/contracts/interfaces/ILendingPool.sol	b028751e2eba3734027e565fe372c9324aaebabd46301497cced0f624a2c2a0a
ILA	projects/vinci-protocol/contracts/interfaces/ILendingPoolAddressesProvider.sol	80e38a1b5569888b4e6c69df6afef578a7a9a347931b4aedd3a45f8b8757a805

ID	File	SHA256 Checksum
ILR	projects/vinci-protocol/contracts/interfaces/ILendingPoolAddressesProviderRegistry.sol	772e7990405cefed09159b59f09b2f306cdd8d54d0ee7727769d832c70d87286
ILC	projects/vinci-protocol/contracts/interfaces/ILendingPoolCollateralManager.sol	8a990d896c6adaed8efd884d6d76fc1d65fd8c132cd36932297c0a72daf41af8
ILK	projects/vinci-protocol/contracts/interfaces/ILendingPoolConfigurator.sol	ad4ee54d0c54c830fbbeaf05e4e18a056cf65ea31bc2bf63d9a83e94c5f64f78
ILO	projects/vinci-protocol/contracts/interfaces/ILendingRateOracle.sol	1c597a8f9e823d2d4cd7dcb07d918a0ccbbfacecc7c877a8c99fa5bf894785ca
INF	projects/vinci-protocol/contracts/interfaces/INFTXEligibility.sol	0f40cac6d4d8f08d501a4ed51d853495805910062cae339184661350644aa0b3
INC	projects/vinci-protocol/contracts/interfaces/INToken.sol	24a9c2590ae8b93bfe614bdd477cda7853f23165708d40a132f1541daf04619e
IPS	projects/vinci-protocol/contracts/interfaces/IParaSwapAugustus.sol	33b9d8e3e4599fd0f0d28eae0f7cd12f2d9dcce178b7a6f99a0c29cf43e787b4
IPA	projects/vinci-protocol/contracts/interfaces/IParaSwapAugustusRegistry.sol	175507ed264112d5daa696225d101a82506238b4990c5617b3f598f769faef1e
IPN	projects/vinci-protocol/contracts/interfaces/IPrevNftxContract.sol	dc2ca7cb84854c1509c383a7106e3ace8080e525c4e5078011b3d0fff9659fce
IPO	projects/vinci-protocol/contracts/interfaces/IPriceOracle.sol	278c8800d826049fbf5d55a6f25fa29afa070d44be5fb24efd79f95b30454b1e
IPG	projects/vinci-protocol/contracts/interfaces/IPriceOracleGetter.sol	e97bcc5c7a92d9b9ab04db1b36ab08d6b473fb968c1c928b9a4e43dae3b50ca7
IRI	projects/vinci-protocol/contracts/interfaces/IReserveInterestRateStrategy.sol	25a776e535f3c19f6e7a344ec3a3c4054624b1f731606a0f8c18e352235f754f
ISB	projects/vinci-protocol/contracts/interfaces/IScaledBalanceToken.sol	70a3ec898f2d986ba3cb9b4e6bd78f1291fba2364066d22852fdb917ca430e79
ISD	projects/vinci-protocol/contracts/interfaces/IStableDebtToken.sol	4a6994c4b7ce7dc59c097cca96c0da68afeed97d812a3a575369081490499119
ITL	projects/vinci-protocol/contracts/interfaces/ITimeLockableERC721.sol	230858f69bfa28be423e9f9b17fb6b4527d7cf39e58a6e74cb40a322439084ee
IUE	projects/vinci-protocol/contracts/interfaces/IUniswapExchange.sol	0c72fc1096cf9b24c31fc1f6fe84b636d24f2cef6b97689952b1eb9c74abb798

ID	File	SHA256 Checksum
IUV	projects/vinci-protocol/contracts/interfaces/IUniswapV2Router02.sol	66328485067267e8ebb08b9f51e6599ce98ca72bd6514094eea28b310050d5a2
IVT	projects/vinci-protocol/contracts/interfaces/IVToken.sol	315ea17b0b6ece89573e705a13aed98073389be9f04dce67bf0a91f18aa38370
IVD	projects/vinci-protocol/contracts/interfaces/IVariableDebtToken.sol	42388b482899d79aa8da64dd7c5c47064141fa6fb46a06578554f492fb6c8bc2
IED	projects/vinci-protocol/contracts/misc/interfaces/IERC20DetailedBytes.sol	29ebb4502de91097f7efe26b6a5a92bba9ca99a4d8b86bc5843d9544921733fc
IUP	projects/vinci-protocol/contracts/misc/interfaces/IUiPoolDataProvider.sol	1c8de303373b9c5de5343d55aa1f9fdb386fb3b637cbc29dadb0ef988ba005f9
IUR	projects/vinci-protocol/contracts/misc/interfaces/IUniswapV2Router01.sol	7d8c340dbeeb8bdb300ad4f7970e94582caec92174dd2d891643108c5903055f
IUC	projects/vinci-protocol/contracts/misc/interfaces/IUniswapV2Router02.sol	6b598571d006d2c13983c4b20dfa04fea21428e9b132f6b4f16c5c89fa19f008
IWE	projects/vinci-protocol/contracts/misc/interfaces/IWETH.sol	c91bb6e4eece6b37e7c882c29a5fe5f9fad5853383ad4f4a739eaba2ef534b04
IWT	projects/vinci-protocol/contracts/misc/interfaces/IWETHGateway.sol	2be57d54b016bce04fb2528305dd51b2d270b884d26b45f86547da5a2bff009b
ACC	projects/vinci-protocol/contracts/misc/AaveCollector.sol	cdefc8ee9067bc236fc2fac041e6755f69a2d977996c3817dbfb3043aca3c8d9
AOC	projects/vinci-protocol/contracts/misc/AaveOracle.sol	b3bda192f5914e0234dfc0c771866775f16422659523cac073118f8d9a1c5aee
APD	projects/vinci-protocol/contracts/misc/AaveProtocolDataProvider.sol	5cd4e8abaad1a59ff93e41b93d131a5c71f9f569788c72a3b4853da53c8f7601
UPD	projects/vinci-protocol/contracts/misc/UiPoolDataProvider.sol	4cc178849e02a1e3de1bfbeaa6272ce3275cf52e4ed5d79bfe4834963c16df46
WEH	projects/vinci-protocol/contracts/misc/WETHGateway.sol	b52c9d3e609b0c8b512a87cd3145d7b37adb0324ee61c3267ab333963f58d452
WBP	projects/vinci-protocol/contracts/misc/WalletBalanceProvider.sol	1694c55a0631bab653e8b8961b841de8de142a2b85f3f17f74da76ea46ac94d3
LPA	projects/vinci-protocol/contracts/protocol/configuration/LendingPoolAddressesProvider.sol	0a8f3b8a53bfc693b1d904c96850e5b9b1324b00fe8fcb72f6707bed0743f4e6

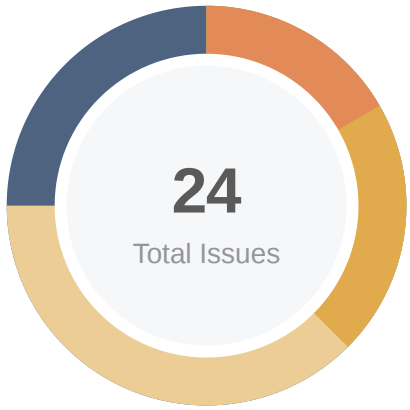
ID	File	SHA256 Checksum
LPP	projects/vinci-protocol/contracts/protocol/configuration/LendingPoolAddressesProviderRegistry.sol	13cec12992873ecdddd8c863fe1420b067538399fe23e9fa2a401b4efa77aca7
DRI	projects/vinci-protocol/contracts/protocol/lendingpool/DefaultReserveInterestRateStrategy.sol	278baff7dca9a739ca8a5b6dd5db6f313ac5984f88663bc6eae94c0ef506ee24
LPC	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPool.sol	ff7e7448f036cd4975cdaae284b6194e1c5026de223939ec1c30e27d09a2139f
LPM	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPoolCollateralManager.sol	1db859555296e4aced63ea82b3cbe3693c45bfdfdeb4ee3649e1f26063efe17f
LPK	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPoolConfigurator.sol	353fb0beaf3f3792d327fd74270908019de3e90752cbae262e16086639d86178
LPS	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPoolStorage.sol	9a03da0b7553c582665652e0b0b41eb79dfc6a3dc93cfabe6ea2728871dedc9c
BIA	projects/vinci-protocol/contracts/protocol/libraries/aave-upgradeability/BaseImmutableAdminUpgradeabilityProxy.sol	3878206082ce0142e5e16c32d5681d6037ccc499f9701293a65ced7b54cb41b9
IIA	projects/vinci-protocol/contracts/protocol/libraries/aave-upgradeability/InitializableImmutableAdminUpgradeabilityProxy.sol	5cc168d72e26405e7a21a84020199696c1ae4d746458b58176316de0debb74de
VIC	projects/vinci-protocol/contracts/protocol/libraries/aave-upgradeability/VersionedInitializable.sol	d8c186700952c66fce1f0c297acf199af72a802af17303debb984928da82f639
NFV	projects/vinci-protocol/contracts/protocol/libraries/configuration/NFTVaultConfiguration.sol	0bdc5d28375f747cf28ce398667d799d54a291a3d8f24cd5549c4a4c290e3f1c
RCC	projects/vinci-protocol/contracts/protocol/libraries/configuration/ReserveConfiguration.sol	c1e92d70ff5b50fb5454457b5e3a8e6b474b63310bee0ca71aa9080fc6c5fa9b
UCC	projects/vinci-protocol/contracts/protocol/libraries/configuration/UserConfiguration.sol	334fa78719d8e17d7dbfeac644c065fd51b50f7123f03ed11413a9e3e5ae6b7f
CUC	projects/vinci-protocol/contracts/protocol/libraries/helpers/ContextUpgradeable.sol	ae710b09a325c1eb34e5624e80be85cd2f52af5f637f2a682023261be2612cff
EKP	projects/vinci-protocol/contracts/protocol/libraries/helpers/Errors.sol	11648e23d52caed7a4768c498df9349786faf3ea8968c08b98cec8a8556e1b76
HCK	projects/vinci-protocol/contracts/protocol/libraries/helpers/Helpers.sol	dec11d52ef906c2f5858e33082b2529a6299130daa60f743cd0a07b116d18cb0

ID	File	SHA256 Checksum
OUC	projects/vinci-protocol/contracts/protocol/libraries/helpers/OwnableUpgradeable.sol	f69290c5c6640db264da0548f4353aa297fb587d84d640445748474ede3f5901
SER	projects/vinci-protocol/contracts/protocol/libraries/helpers/SafeERC721.sol	851317d520479b9239d11806a9adf600f0edb0417a7ae04f3a8f3253bb79632a
NFX	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXAllowAllEligibility.sol	b4d332abc48b15b2298e50976799b8a62bee8120b34fe3e3e9795ffadbce324b
NFA	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXAvastarRank60Eligibility.sol	479ee745de8a06a05eb08834646cdc481bdffbb4a91757e80dde98ef158d1cf2
NFD	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXDeferEligibility.sol	0860073b540ec5448fed06413d03a820bd306c8503749528f349ec14f8194872
NFE	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXDenyEligibility.sol	34d6f48c4a328d13e60e7ac0abdc011add77210890113121398ee6f049b626f0
NFC	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXEligibility.sol	810be63b0d86b7f0680350469abfe1c2b241652b7a1c8e4556bf7dbbab12059c
NFG	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXGen0Eligibility.sol	cc5d042c8f1843745d3ac1e5c09cd598003ba32b97af2320d93cde9877fd77ab
NFF	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXGen0FastEligibility.sol	ea27001751cebf07120d7ec82ca375d22ae7533b36aeddc68649be79815c7c3b
NFK	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXListEligibility.sol	2d1250621e2e9845a84676832d832e519d26b7f9c5d9e1fdcb9866b5af82766d
NFO	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXOpenseaEligibility.sol	9f504fe1b1b178aea3a7e61b0fa8daa57e307c6dc97ea1ab56f2aefff0c2d826
NFR	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXRangeEligibility.sol	5db5602438a5a11a260de70ee9e68ad719b306d9d81c4025d6ab2c6f33539931
NFP	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXRangeExtendedEligibility.sol	e99191fd941ef983ee7497a8f33ce8b2fcad55030fa602346ea7de3825167d2d
NFU	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXUglyEligibility.sol	65b6c795a54f50839feac312fe4c30632aa39a03421d6ded9777a544e369cb1b
NTX	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXUniqueEligibility.sol	a1b74d98e21f15d98db708bf4344b888753d01b3b470d73d0166e6e27d27d98d

ID	File	SHA256 Checksum
UEC	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/UniqueEligibility.sol	a670adcf648539297b3ee5b70d980878fa7bfedba521f84e3812e03a13a66c2d
UVS	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/UniswapV3SparkleEligibility.sol	01603b8db6974fa97f8416ddef705fa523544447a71908515c5d07c9656c77de
GLC	projects/vinci-protocol/contracts/protocol/libraries/logic/GenericLogic.sol	cd9ec6609ba4408e6d37a5355e5a7042e5747ce1ac079c6f7ccce94a47694651
NFL	projects/vinci-protocol/contracts/protocol/libraries/logic/NFTVaultLogic.sol	9596375ded3dfec5079c41915c7e055a43e470744b6789a129635b9927219ff8
RLC	projects/vinci-protocol/contracts/protocol/libraries/logic/ReserveLogic.sol	bf1c6f73c2499a9506c1d855206b52830bc7633eeba566b54e648469f4f932ec
VLC	projects/vinci-protocol/contracts/protocol/libraries/logic/ValidationLogic.sol	634d2d03a321a7ab2cba07d51b80ffc4ac524c9852e1d430717616cf9a494702
MUC	projects/vinci-protocol/contracts/protocol/libraries/math/MathUtils.sol	ccd1165d1f6fd6f41fcf547a0dc514c29ed00c0a843f04e8a72196dece00888c
PMC	projects/vinci-protocol/contracts/protocol/libraries/math/PercentageMath.sol	959091d7b378e342d2523646317fefd67dac0f0b2561100f93d2d3a1621e91c6
WRM	projects/vinci-protocol/contracts/protocol/libraries/math/WadRayMath.sol	981267e56da3685d20d682ac5e61f9920bf9aae73e342fd12506807530de8e5c
DTC	projects/vinci-protocol/contracts/protocol/libraries/types/DataTypes.sol	6be55912f6474e553ea43fe6f251d42af0f2ceb21f479357056f194ff5fcdc50
DTB	projects/vinci-protocol/contracts/protocol/tokenization/base/DebtTokenBase.sol	6d8d53884eefaaa2a94b99533e4d06d3dc289b9fc6627e60769ad0bc77573e99
DAV	projects/vinci-protocol/contracts/protocol/tokenization/DelegationAwareVToken.sol	9ce85bfc91cbc99a9aac9988377d8dc3892686738461ff31bb80d22fdc66d4f0
IRC	projects/vinci-protocol/contracts/protocol/tokenization/IncentivizedERC20.sol	0f39ae2ee2110737bee0ad6a2c22906a06d12fbac0f9f3a20610281d719f9cb6
NTC	projects/vinci-protocol/contracts/protocol/tokenization/NToken.sol	f54eff35447c0cdb40da431480f0e8949b2ff6eb28eddb544cafcc49dd7b1734
TLN	projects/vinci-protocol/contracts/protocol/tokenization/TimeLockableNToken.sol	4c6ba2cad01ce75dd332d8549fd23b5c2d4555570f8aeab542a5b4d426e39a5f

ID	File	SHA256 Checksum
TLT	projects/vinci-protocol/contracts/protocol/tokenization/TimeLockableNTokenForTest.sol	203cfa01847e6241ae248a9df63716aa160ec76d08403b8dd82ad7da0eea5c7c
VTC	projects/vinci-protocol/contracts/protocol/tokenization/VToken.sol	c21d302ee3c4d18ee5086e81f392def173fd2d9c7ba996faba76ce9eb227590b
VDT	projects/vinci-protocol/contracts/protocol/tokenization/VariableDebtToken.sol	7fc63e31bd988162d571d830e47a301777dc19b01e1ef37f9e547944ebb0529e
WER	projects/vinci-protocol/contracts/protocol/tokenization/WrappedERC721.sol	d73ac4826fa193be6f0bba0c580db2fc4afe1072b4ecd8818108deadc128463b

Findings



Critical	0 (0.00%)
Major	4 (16.67%)
Medium	5 (20.83%)
Minor	9 (37.50%)
Informational	6 (25.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
CKP-01	Third Party Dependency In Multiple Contracts	Volatile Code	Minor	Acknowledged
CON-01	Centralization Risks In LendingPoolAddressesProvider And LendingPoolAddressesProviderRegistry	**Centralization / Privilege**	**Major**	Mitigated
ELI-01	Centralization Risks In Eligibility Contracts	**Centralization / Privilege**	**Major**	Acknowledged
ELI-02	Third Party Dependencies In NFT Eligibility Check	Volatile Code	Minor	Acknowledged
GLC-01	Redundant Code In `GenericLogic.calculateUserAccountData`	Logical Issue	Minor	Acknowledged
LEN-01	Potential Reentrancy Through Token Transfer And Minting	Logical Issue	Medium	Resolved
LPC-01	No Fee For Flashloan	Inconsistency, Volatile Code	Minor	Acknowledged
LPC-02	Ambiguous Error Message	Volatile Code	Informational	Resolved
LPC-03	Redundant Variable `referralCode`	Logical Issue	Informational	Acknowledged
LPC-04	Missing Emit Events	Volatile Code	Informational	Acknowledged
LPK-01	Centralization Related Risks In LendingPoolConfigurator	**Centralization / Privilege**	**Major**	Mitigated
LPK-02	Using Wrong Function To Check For Liquidity	Logical Issue	Medium	Resolved

ID	Title	Category	Severity	Status
LPM-01	Logic Of Function `nftLiquidationCall`	Logical Issue	Medium	Resolved
LPM-02	`setUsingNFTVaultAsCollateral` Bypassed	Logical Issue	Medium	Resolved
LPM-03	Ownership Of NFT Token Not Checked	Control Flow	Minor	Resolved
LPM-04	Check Effect Interaction Pattern Violated	Logical Issue	Minor	Acknowledged
MIS-01	Centralization Risks In Misc Contracts	**Centralization / Privilege**	**Major**	Mitigated
NTC-01	Inconsistent Amount Check In `burn` And `burnBatch`	Inconsistency	Medium	Resolved
NTC-02	Recommend To Add Ownership Check When Burning NTokens	Logical Issue	Informational	Acknowledged
NTC-03	Compatibility Issue With ERC1155 In Contract `NToken`	Logical Issue	Informational	Acknowledged
PRO-01	NFT Lock Not Implemented	Logical Issue, Inconsistency	Minor	Acknowledged
PRO-02	Shadowing Built-In Symbol	Coding Style	Informational	Acknowledged
VLC-01	Array Length Not Checked Before Loop	Volatile Code	Minor	Acknowledged
WBP-01	Locked Ether In Contract WalletBalanceProvider	Language Specific	Minor	Resolved

CKP-01 | Third Party Dependency In Multiple Contracts

Category	Severity	Location	Status
Volatile Code	Minor	projects/vinci-protocol/contracts/misc/AaveOracle.sol (1): 26; projects/vinci-protocol/contracts/misc/UiPoolDataProvider.sol (1): 34, 35; projects/vinci-protocol/contracts/misc/WETHGateway.sol (1): 20; projects/vinci-protocol/contracts/protocol/configuration/LendingPoolAddressesProvider.sol (1): 167~169; projects/vinci-protocol/contracts/protocol/lendingpool/LendingPool.sol (1): 1020~1023; projects/vinci-protocol/contracts/protocol/lendingpool/LendingPoolCollateralManager.sol (1): 272~274; projects/vinci-protocol/contracts/protocol/tokenization/VToken.sol (1): 43	Acknowledged

Description

The linked contracts are serving as the underlying entity to interact with one or more third party protocols. The scope of the audit treats third party entities as black boxes and assume their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties, migrating to new LP pools, etc.

Specifically, the prices for the NFTs and the ERC20 tokens are received from external price oracle, and used for calculating borrow amount, liquidation amount, and risk parameters, etc. Any upgrade/change in the external price oracle may affect the functionality of this project as well.

Other dependencies are listed as follows:

- The contract `AaveOracle` interacts with third party contract with `IChainlinkAggregator` interface via `assetsSources`.
- The contract `UiPoolDataProvider` interacts with third party contract with `IChainlinkAggregator` interface via `networkBaseTokenPriceInUsdProxyAggregator`.
- The contract `UiPoolDataProvider` interacts with third party contract with `IChainlinkAggregator` interface via `marketReferenceCurrencyPriceInUsdProxyAggregator`.
- The contract `WETHGateway` interacts with third party contract with `IWETH` interface via `WETH`.
- The contract `VToken` interacts with third party contract with `IDelegationToken` interface via `_underlyingAsset`.

Recommendation

We understand that the business logic requires interaction with these third parties. We encourage the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

Alleviation

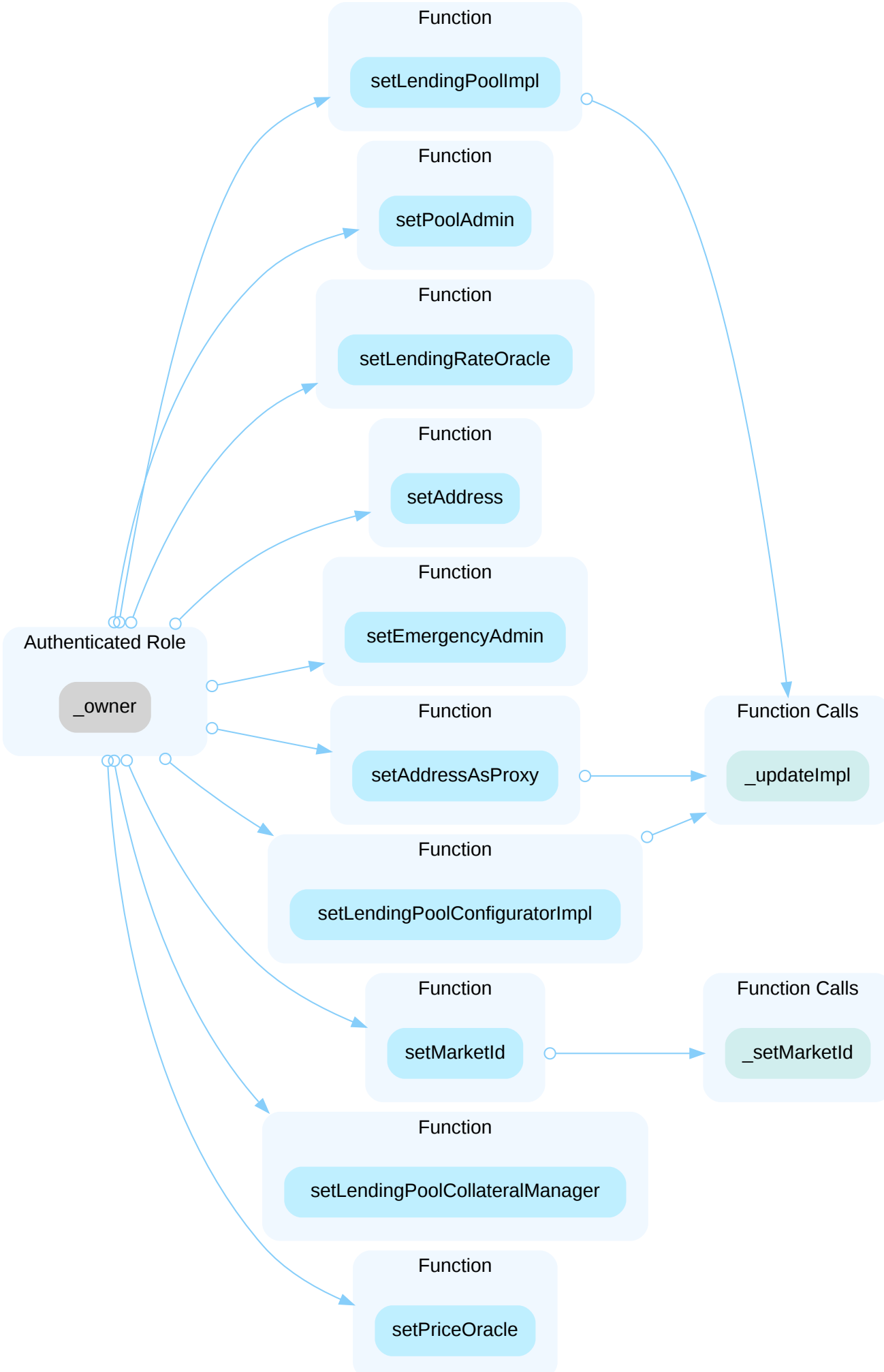
[Vinci Team]: Issue acknowledged. I won't make any changes for the current version.

CON-01 | Centralization Risks In LendingPoolAddressesProvider And LendingPoolAddressesProviderRegistry

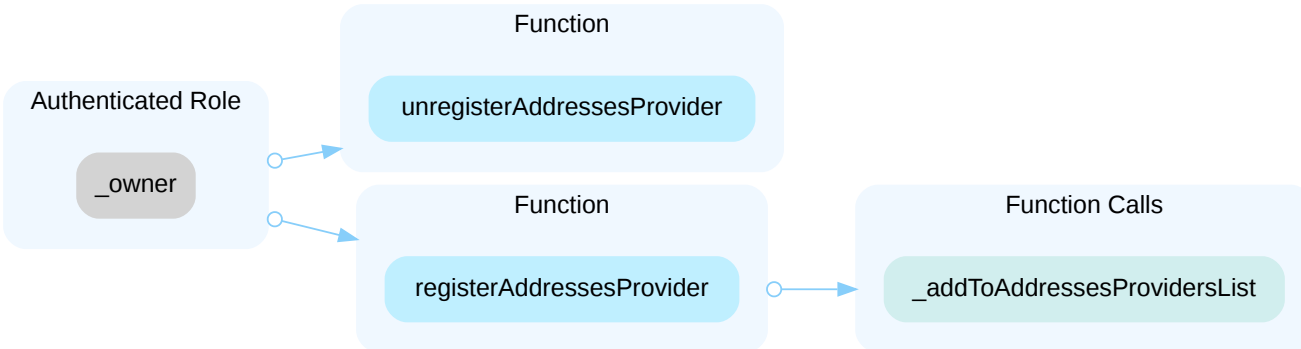
Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/vinci-protocol/contracts/protocol/configuration/LendingPoolAddressesProvider.sol (1): 47, 60, 75, 101, 119, 139, 153, 162, 171, 180; projects/vinci-protocol/contracts/protocol/configuration/LendingPoolAddressesProviderRegistry.sol (1): 47, 59	Mitigated

Description

In the contract `LendingPoolAddressesProvider` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority, set the lending pool implementation, market, price oracle, and emergency admin.


Function
setLendingPoolImpl
Function
setPoolAdmin
Function
setLendingRateOracle
Function
setAddress
Function
setEmergencyAdmin
Authenticated Role
_owner
Function
setAddressAsProxy
Function Calls
_updateImpl
Function
setLendingPoolConfiguratorImpl
Function
setMarketId
Function Calls
_setMarketId
Function
setLendingPoolCollateralManager
Function
setPriceOracle

In the contract `LendingPoolAddressesProviderRegistry` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and change the lending pool addresses provider.



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., multisignature 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.

Alleviation

The team acknowledged the issue and adopted the timelock and multi-sig solution to ensure the private key management process at the current stage. The `LendingPoolAddressesProvider` and `LendingPoolAddressesProviderRegistry` contracts have transferred the ownership to a timelock contract, whereas the timelock contract's admin privilege is transferred to a Gnosis Safe contract with 3/5 signers in the sensitive function signing process.

[Vinci Team]:

The owner transfer transaction hash of LendingPoolAddressesProviderRegistry is:

https://etherscan.io/tx/0xbdbaf5c72b7cbd2f0b16f0d57b18136bf96b450999c2fe3e5abf31060f0a8375

Multi-Signature

Multi-sign proxy address: https://etherscan.io/address/0x78573a38f34b2f6eab5c21b33bd678afa8c0c7af

Transaction proof for transferring ownership to multi-signature proxy:
https://etherscan.io/tx/0x2369e7ae28ad02d3ad444d30faf5eba59cf58657e4122fbfbbcd225031e6348f

Internal multi-signature address:
https://etherscan.io/address/0xEB6D25d3FA2fe832DCe54C284368dCcbDE56F8Ed,https://etherscan.io/address/0xB9b1d21Cd5cc4EAb58f1cDB6C644AbDf10A275eB,https://etherscan.io/address/0x6613Be835bB6

6425EcE1A1268a1D5DD3cbD13519,https://etherscan.io/address/0xed1639c1f0ce914861031CfA0Ca0Fa10D5819b6e,https://etherscan.io/address/0x2c4ce891e4BD5fb59E1a8101Aa8524174E8A1Df7

Time-lock

Time lock contract address: https://etherscan.io/address/0x78573a38f34b2f6eab5c21b33bd678afa8c0c7af

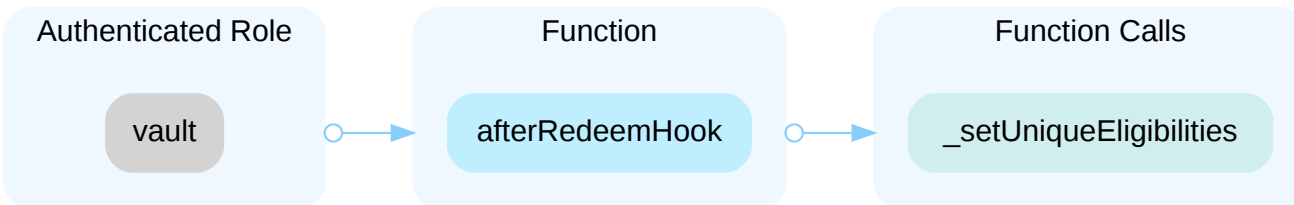
Time lock owner transfer transaction hash:
https://etherscan.io/tx/0x9427cb150f03f3eb92bf11c4d27e9294d065f999f9738c1aa0478441b5475621

ELI-01 | Centralization Risks In Eligibility Contracts

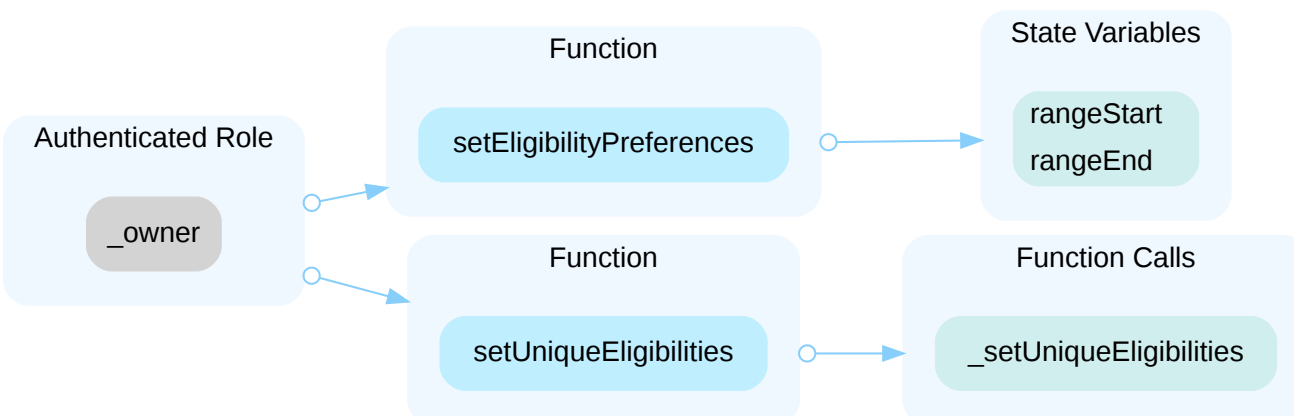
Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXDenyEligibility.sol (1): 24; projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXRangeExtendedEligibility.sol (1): 75, 86; projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXUniqueEligibility.sol (1): 95, 103, 111, 122; projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/UniswapV3SparkleEligibility.sol (1): 137	Acknowledged

Description

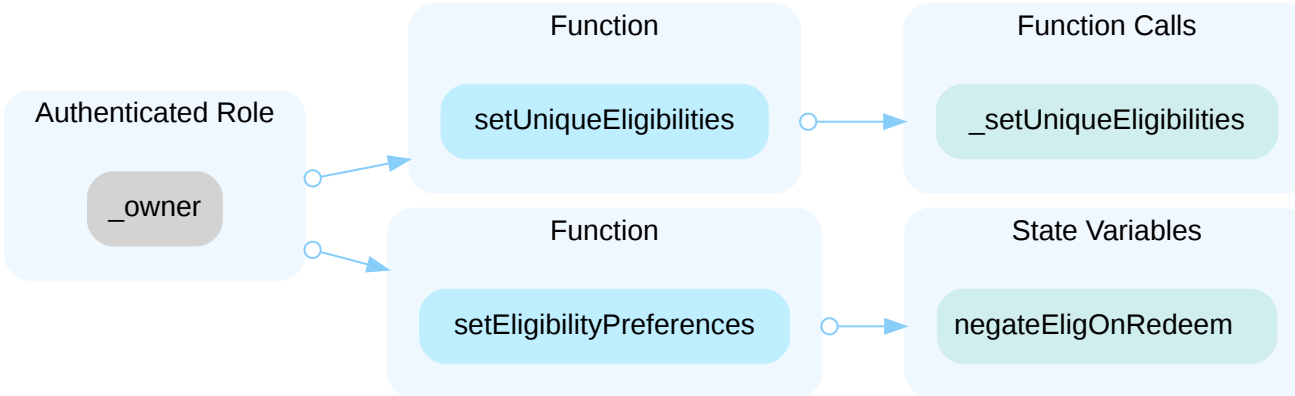
In the contract `NFTXDenyEligibility` the role `vault` has authority over the functions shown in the diagram below. Any compromise to the `vault` account may allow the hacker to take advantage of this authority and set the eligibility.



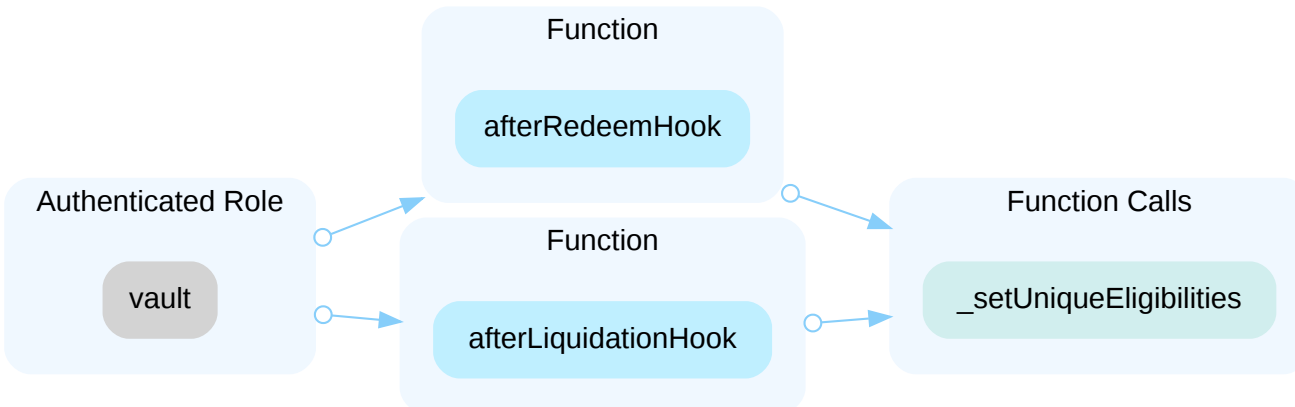
In the contract `NFTXRangeExtendedEligibility` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set the eligibility.



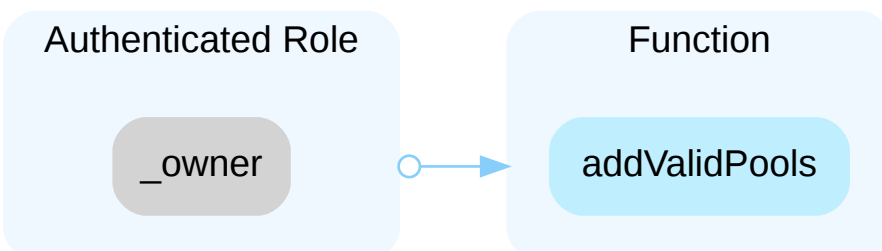
In the contract `NFTXUniqueEligibility` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set the eligibility.



In the contract `NFTXUniqueEligibility` the role `vault` has authority over the functions shown in the diagram below. Any compromise to the `vault` account may allow the hacker to take advantage of this authority and set the eligibility.



In the contract `UniswapV3SparkleEligibility` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and add valid pool.



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., multisignature 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.

Alleviation

[Vinci Team]:

The _owner will be set as the address of LendingConfigurator.

The vault will be set as the address of LendingPool after deployment.

ELI-02 | Third Party Dependencies In NFT Eligibility Check

Category	Severity	Location	Status
Volatile Code	Minor	projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXAvastarRank60Eligibility.sol (1): 58; projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXDeferEligibility.sol (1): 27; projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXGen0Eligibility.sol (1): 65~66; projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXGen0FastEligibility.sol (1): 47; projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/NFTXUglyEligibility.sol (1): 51~52; projects/vinci-protocol/contracts/protocol/libraries/logic/eligibility/UniswapV3SparkleEligibility.sol (1): 147~148, 153~156	Acknowledged

Description

The linked contracts are serving as underlying entities to interact with multiple third party protocols, mainly NFT projects, for eligibility validation. Details are listed below:

- The contract `NFTXAvastarRank60Eligibility` interacts with third party contract with `KittyCore` interface
- The contract `NFTXGen0KittyEligibility` and `NFTXGen0FastKittyEligibility` interacts with third party contract with `Avastar` interface
- The contract `NFTXUglyEligibility` interacts with third party contract with `IPolymorph` interface
- The contract `UniswapV3SparkleEligibility` interacts with third party contract with `INonfungiblePositionManager` interface
- The contract `NFTXDeferEligibility` interacts with third party contract with `IPrevNftxContract` interface
- The contract `NFTXDeferEligibility` interacts with external contract `deferAddress`

The scope of the audit treats third party entities as black boxes and assume their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties, migrating to addresses, etc.

Recommendation

We understand that the business logic of Vinci requires interaction with third party NFT contracts and external oracle contracts. We encourage the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[Vinci Team]: Issue acknowledged. I won't make any changes for the current version.

GLC-01 | Redundant Code In `GenericLogic.calculateUserAccountData`

Category	Severity	Location	Status
Logical Issue	Minor	projects/vinci-protocol/contracts/protocol/libraries/logic/GenericLogic.sol (1) : 184~195	Acknowledged

Description

Line 184 to 195 in library `GenericLogic` calculates the total collateral value for the ERC20 tokens the user deposited. The *for* loop will only execute if `userConfig.isUsingAsCollateral(vars.i)` returns True. However, the current audit didn't locate any code that could set ERC20 as collateral, which means that the code in this if branch will never be executed.

```
184        if (vars.liquidationThreshold != 0 && userConfig.isUsingAsCollateral(vars.i))
{
185          vars.compoundedLiquidityBalance =
IERC20(currentReserve.vTokenAddress).balanceOf(user);
186     ...
```

Recommendation

We recommend to remove the mentioned code snippet if ERC20 is not considered collateral in project design.

Alleviation

[Vinci Team] This is reserved for future business.

LEN-01 | Potential Reentrancy Through Token Transfer And Minting

Category	Severity	Location	Status
Logical Issue	Medium	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPool.sol (1): 216, 252, 319, 531; projects/vinci-protocol/contracts/protocol/lendingpool/LendingPoolCollateralManager.sol (1): 176, 187~193	Resolved

Description

The order of external call/transfer and storage manipulation must follow the check-effect-interaction pattern. The ERC721 safeTransferFrom(), safeTransfer() and safeMint() functions have an external call _checkOnERC721Received() to the recipient, which might cause potential reentrancy.

Recommendation

We advise the client to check if storage manipulation is before the external call/transfer operation.LINK

Alleviation

[Vinci Team]:

Fixed in commit 81c20ce402376fc3e622243ec609cfdb475ca4d1.

A reentrancy with UsingNFTAsCollateral being false and the attacker having any debt will cause him to be unable to withdraw/transfer/borrow assets. Impact on this reentrancy is negligible.

LPC-01 | No Fee For Flashloan

Category	Severity	Location	Status
Inconsistency, Volatile Code	Minor	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPool.sol (1): 502, 524, 538~540	Acknowledged

Description

The code in function `nftFlashLoan()` allows the users to flash loan the borrower's deposited NFT without any fee. Still, the comment declares that `as long as the amount taken plus a fee is returned`, and `_flashLoadPremiumTotal` is not used.

Recommendation

We recommend to confirm if the current implementation and comment aligns with the original project design.

Alleviation

[Vinci Team]:

This mismatch is due to business logic changes. We may add back the ability to charge a fee for flash-loan in the future by updating the logic contract.

LPC-02 | Ambiguous Error Message

Category	Severity	Location	Status
Volatile Code	Informational	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPool.sol (1): 1092, 1107	Resolved

Description

Line 1092 and 1107 of contract `LendingPool` returns the same error ID for the upper limit checks of reverse and NFT vault. This makes it difficult to locate the error.

Recommendation

We advise the client to implement separate error IDs for different checks.

Alleviation

[Vinci Team]: Issue acknowledged. Changes have been reflected in the commit hash 26bc18ed829d8bda3f0824c2501b3cc99fc2ba2c

LPC-03 | Redundant Variable `referralCode`

Category	Severity	Location	Status
Logical Issue	Informational	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPool.sol (1): 517	Acknowledged

Description

In contract `LendingPool`, `referralCode` is required for multiple functions, while the code does not validate or user it except for emitting the event.

Recommendation

Recommend to remove the redundant variable if there's no related business logic.

Alleviation

[Vinci Team]:

It works as intended. We will collect referral codes from events for future use.

LPC-04 | Missing Emit Events

Category	Severity	Location	Status
Volatile Code	Informational	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPool.sol (1): 918~929, 942, 956, 970, 978, 986	Acknowledged

Description

The sensitive functions' input parameters miss emitting the event logs:

- initNFTVault
- setReserveInterestRateStrategyAddress
- setConfiguration
- setNFTVaultConfiguration
- setNFTVaultActionExpiration
- setNFTVaultEligibility

Recommendation

We recommend emitting the event logs for the functions which changes the statements.

LPK-01 | Centralization Related Risks In LendingPoolConfigurator

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPoolConfigurator.sol (1): 39, 66, 150, 205, 236, 270~272, 307, 330, 349, 375~377, 393, 411~416, 459, 473, 487, 501, 518, 532, 547, 562~564, 574, 574, 588, 605, 615, 624	Mitigated

Description

The `PoolAdmin` of the contract `LendingPoolConfigurator` has the responsibility to notify users about the following capabilities:

- init reserves via `batchInitReserve`
- control NFT vaults via `batchInitNFTVault`, `updateNToken`, `activateNFTVault` `freezeNFTVault` and `updateNFTVaultActionExpiration`
- update VToken via `updateVToken`
- update debt tokens via `updateVariableDebtToken` and `deactivateNFTVault`
- change NFT eligibility via `updateNFTEligibility` and `setNFTEligibility`
- set the borrowing and collateral via `enableBorrowingOnReserve` `disableBorrowingOnReserve` `configureNFTVaultAsCollateral`
- set the reserve via `enableReserveStableRate` `disableReserveStableRate` `activateReserve` `deactivateReserve` `freezeReserve` `unfreezeReserve` and `setReserveFactor`
- set the interest rate strategy via `setReserveInterestRateStrategyAddress` Any compromise to the `PoolAdmin` account may allow the hacker to take advantage of this authority.

The `onlyEmergencyAdmin` of the contract `LendingPoolConfigurator` has the responsibility to notify users about the following capabilities:

- pause the pool via `setPoolPause` Any compromise to the `onlyEmergencyAdmin` account may allow the 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., multisignature 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

Alleviation

The team acknowledged the issue and adopted the timelock and multi-sig solution to ensure the private key management process at the current stage. The `LendingPoolConfigurator` contract has transferred the

ownership to a timelock contract, whereas the timelock contract's admin privilege is transferred to a Gnosis Safe contract with 3/5 signers in the sensitive function signing process.

[Vinci Team]:

Multi-signature

Multi-sign proxy address: https://etherscan.io/address/0x78573a38f34b2f6eab5c21b33bd678afa8c0c7af

Transaction proof for transferring ownership to multi-signature proxy: https://etherscan.io/tx/0x2369e7ae28ad02d3ad444d30faf5eba59cf58657e4122fbfbbcd225031e6348f

Internal multi-signature address: https://etherscan.io/address/0xEB6D25d3FA2fe832DCe54C284368dCcbDE56F8Ed,https://etherscan.io/address/0xB9b1d21Cd5cc4EAb58f1cDB6C644AbDf10A275eB,https://etherscan.io/address/0x6613Be835bB66425EcE1A1268a1D5DD3cbD13519,https://etherscan.io/address/0xed1639c1f0ce914861031CfA0Ca0Fa10D5819b6e,https://etherscan.io/address/0x2c4ce891e4BD5fb59E1a8101Aa8524174E8A1Df7

Time-lock

Time lock contract address: https://etherscan.io/address/0x78573a38f34b2f6eab5c21b33bd678afa8c0c7af

Time lock owner transfer transaction hash: https://etherscan.io/tx/0x7a22ea54303681acee886f53c0e67202c6c3a480dfcf503df896d5e3cf99bde3

LPK-02 | Using Wrong Function To Check For Liquidity

Category	Severity	Location	Status
Logical Issue	Medium	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPoolConfigurator.sol (1): 443	Resolved

Description

The function `_checkNoLiquidity()` is to check if the asset has liquidity. The input parameter is the address of ERC20, but `nft` is the address of NFT, which causes the function to be reverted.

Recommendation

We advise the client to use `_checkNFTVaultNoLiquidity()` instead of `_checkNoLiquidity()`.

Alleviation

[Vinci Team]: Issue acknowledged. Changes have been reflected in the commit hash 26bc18ed829d8bda3f0824c2501b3cc99fc2ba2c

LPM-01 | Logic Of Function `nftLiquidationCall`

Category	Severity	Location	Status
Logical Issue	Medium	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPoolCollateralManager.sol (1): 129, 296	Resolved

Description

The `maxLiquidatableDebt` defines the maximum 50% of the debtor's debt that can be liquidated in liquidation. However, line 296 compares the object "userTotalDebt", which is the debt of all users. If the function `nftLiquidationCall` follows the current logic, it is possible for the liquidator to liquidate all the user's debts. Once the user's debt is cleared, the borrowing status of `_usersConfig` needs to be updated to false.

Recommendation

We recommend to add a logic to update the borrowing status of `_usersConfig` in the case where all the debt of the borrower is liquidated.

Alleviation

[Vinci Team]:

Issue acknowledged. Changes have been reflected in the commit hash 1843696:

https://github.com/VinciProtocol/vinci-protocol/commit/1843696b155352faa6285354b57abb5471515025

LPM-02 | `setUsingNFTVaultAsCollateral` Bypassed

Category	Severity	Location	Status
Logical Issue	Medium	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPoolCollateralManager.sol (1): 198~201	Resolved

Description

If the collateral being liquidated is equal to the user balance, the function sets the currency as not being used as collateral anymore. However, the balance of the user has been transferred or burned. This does not satisfy the check condition of line 198 and bypasses the setup.

Recommendation

We advise the client to recheck the logic.

Alleviation

[Vinic Team]: Issue acknowledged. Changes have been reflected in the commit hash 26bc18ed829d8bda3f0824c2501b3cc99fc2ba2c

LPM-03 | Ownership Of NFT Token Not Checked

Category	Severity	Location	Status
Control Flow	Minor	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPoolCollateralManager.sol (1): 148	Resolved

Description

Missing check that the owner of `tokenIds` is `user`. This causes the function `_calculateAvailableNFTCollateralToLiquidate()` to return the value `actualDebtToLiquidate` as 0, because the function `getLiquidationAmounts()` detects the owner of `tokenIds`. If the `actualDebtToLiquidate` is 0, liquidation is pointless and a waste of gas.

Recommendation

We advise the client to add a check.

Alleviation

[Vinci Team]: Issue acknowledged. Changes have been reflected in the commit hash 26bc18ed829d8bda3f0824c2501b3cc99fc2ba2c

LPM-04 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Logical Issue	Minor	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPoolCollateralManager.sol (1): 198	Acknowledged

Description

The order of external call/transfer and storage manipulation must follow the check-effect-interaction pattern.

Recommendation

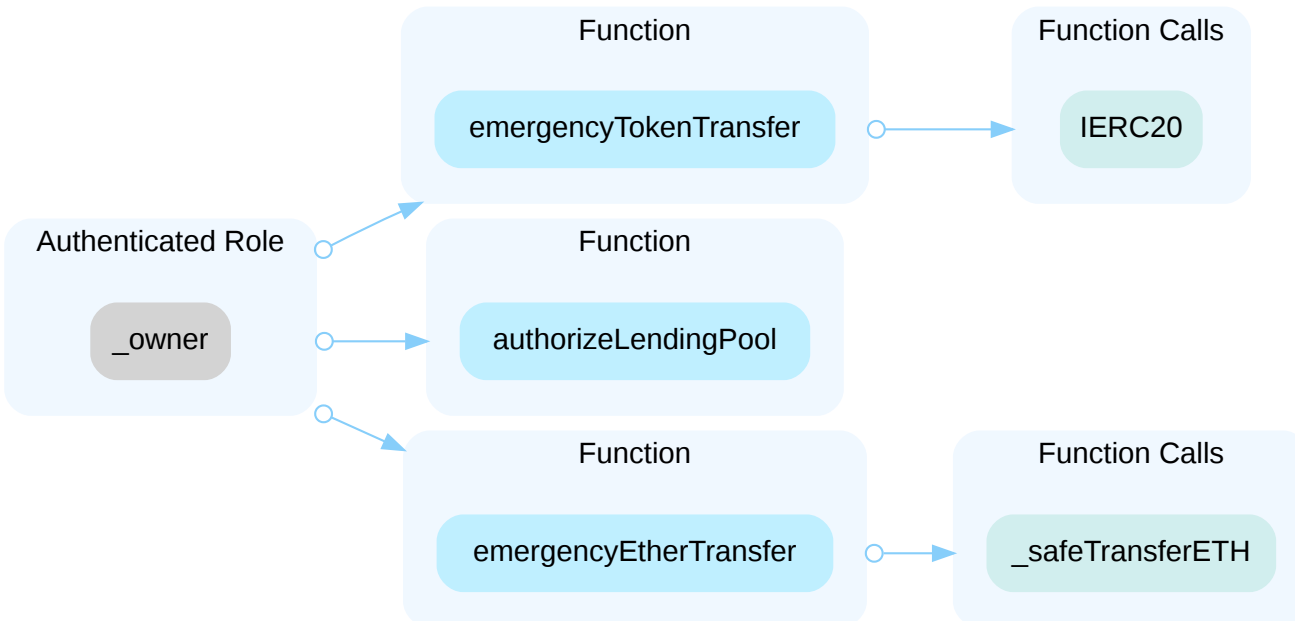
We advise the client to check if storage manipulation is before the external call/transfer operation.LINK

MIS-01 | Centralization Risks In Misc Contracts

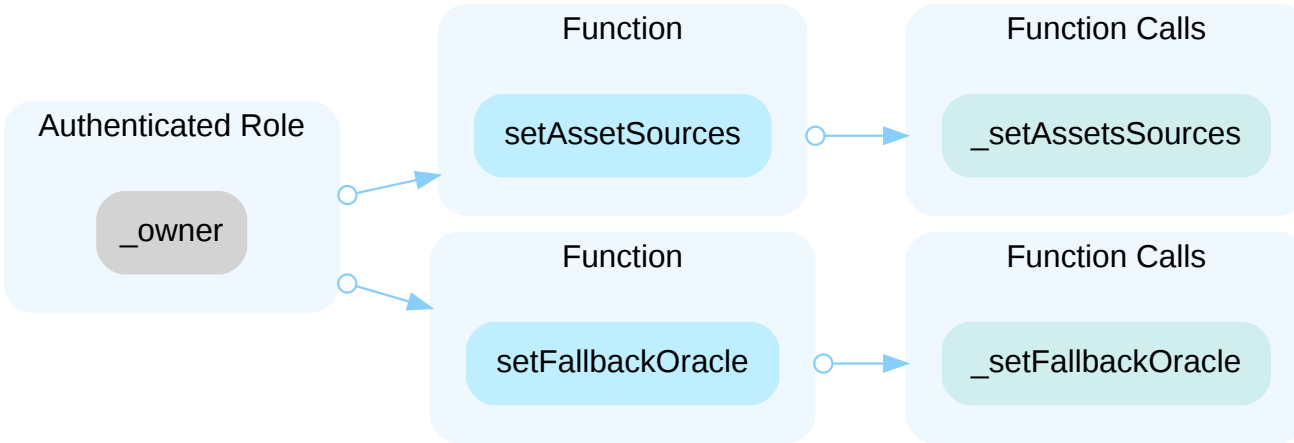
Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/vinci-protocol/contracts/misc/AaveOracle.sol (1): 64, 74; projects/vinci-protocol/contracts/misc/WETHGateway.sol (1): 30, 151, 165	Mitigated

Description

In the contract `WETHGateway` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority, authorize any address as the lending pool, and transfer Token & ETH.



In the contract `AaveOracle` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set the asset sources and fallback oracle.



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., multisignature 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.

Alleviation

The team acknowledged the issue and adopted the timelock and multi-sig solution to ensure the private key management process at the current stage. The `WETHGateway` and `AaveOracle` contracts have transferred the ownership to a timelock contract, whereas the timelock contract's admin privilege is transferred to a Gnosis Safe contract with 3/5 signers in the sensitive function signing process.

[Vinci]:

The owner transfer transaction hash of WPUNKSGateway is:
https://etherscan.io/tx/0x1c50241f09b7e868c100f1fff3e3dec1cd3d44c6d4a6b1fc40462bd0202a630a

The owner transfer transaction hash of WETHGateway is:
https://etherscan.io/tx/0xb5d83ffcfa7ffbc12b7179b75ac68045772cc0b1bca392529e9139a3d6e3b2ec

Multi-signature

Multi-sign proxy address: https://etherscan.io/address/0x78573a38f34b2f6eab5c21b33bd678afa8c0c7af

Transaction proof for transferring ownership to multi-signature proxy:
https://etherscan.io/tx/0x2369e7ae28ad02d3ad444d30faf5eba59cf58657e4122fbfbbcd225031e6348f

Internal multi-signature address:
https://etherscan.io/address/0xEB6D25d3FA2fe832DCe54C284368dCcbDE56F8Ed,https://etherscan.io/address/0xB9b1d21Cd5cc4EAb58f1cDB6C644AbDf10A275eB,https://etherscan.io/address/0x6613Be835bB66425EcE1A1268a1D5DD3cbD13519,https://etherscan.io/address/0xed1639c1f0ce914861031CfA0Ca0Fa10D5819b6e,https://etherscan.io/address/0x2c4ce891e4BD5fb59E1a8101Aa8524174E8A1Df7

Time-lock

Time lock contract address: https://etherscan.io/address/0x78573a38f34b2f6eab5c21b33bd678afa8c0c7af

Time lock owner transfer transaction hash:
https://etherscan.io/tx/0x9adf398c4971ddf36a914844bd2168afa00c8c30490c23c6157e4d93283ff46f

NTC-01 | Inconsistent Amount Check In `burn` And `burnBatch`

Category	Severity	Location	Status
Inconsistency	Medium	projects/vinci-protocol/contracts/protocol/tokenization/NToken.sol (1): 170~178	Resolved

Description

The transaction reverts when amount = 0 in the function `burn`. But `_uderlyingBFT` could be transferred to `receiverOfUnderlying` in the function `burnBatch`.

Function `burn`:

```
155       require(amount != 0, Errors.CT_INVALID_BURN_AMOUNT);
156       _burn(tokenId);
157       IERC721(_underlyingNFT).safeTransferFrom(address(this), receiverOfUnderlying,
tokenId, "");
```

Function `burnBatch`:

```
170       for(uint256 i = 0; i < tokenIds.length; ++i){
171         if(amounts[i] != 0){
172          _burn(tokenIds[i]);
173         }
174       }
175       for(uint256 i = 0; i < tokenIds.length; ++i){
176         uint256 id = tokenIds[i];
177         IERC721(_underlyingNFT).safeTransferFrom(address(this), receiverOfUnderlying,
id, "");
178       }
```

Recommendation

Recommend to make the `burnBatch` logic consistent with `burn` function.

Alleviation

[Vinci Team]: Issue acknowledged. Changes have been reflected in the commit hash 26bc18ed829d8bda3f0824c2501b3cc99fc2ba2c

NTC-02 | Recommend To Add Ownership Check When Burning NTokens

Category	Severity	Location	Status
Logical Issue	Informational	projects/vinci-protocol/contracts/protocol/tokenization/NToken.sol (1): 149, 163	Acknowledged

Description

The function `burn()` and `burnBatch()` does not check if the user is the owner of the `tokenId`, which leads to the possibility of transferring other people's underlying assets.

While this check is not implemented within NToken contract, the validation is actually carried out in LendingPool and LendingPoolCollateralManager, before the function `burnBatch()` is called. Nevertheless, the logic might be clearer and more robust if this check is placed within function `burn()` and `burnBatch()`.

Recommendation

We recommend adding a check that the user is the owner of the `tokenId` within the `burn()` and `burnBatch()` function.

Alleviation

[Vinci Team]: Issue acknowledged. I won't make any changes for the current version.

NTC-03 | Compatibility Issue With ERC1155 In Contract `NToken`

Category	Severity	Location	Status
Logical Issue	Informational	projects/vinci-protocol/contracts/protocol/tokenization/NToken.sol (1): 90	Acknowledged

Description

1. The `NToken` contract could receive ERC1155 tokens, but there are no withdraw function for ERC1155 tokens in its implementation. ERC155 token could be locked inside the contract.
2. The function `setClaimAdmin()` is only called by the lending pool, but it is not called in the contract `LendingPool`. The functions `claimERC20Airdrop()`, `claimERC721Airdrop()` and `claimERC1155Airdrop()` can not be called if the claim admin is not set, because the caller can only be the claim admin.
3. The `permit` function is not implemented.

Recommendation

We recommend to confirm if the current implementation aligns with the original project design.

Alleviation

[Vinci Team]:

NToken's ERC1155Receiver interface is used to obtain Airdrop rewards, which will be extracted through the claimERC1155Airdrop interface.
The claimERC****Airdrop interface is reserved for future LendingPool upgrades to return the Airdrop reward to the user by calling the nftFlashLoan interface.
We will not implement the permit function in this version.

PRO-01 | NFT Lock Not Implemented

Category	Severity	Location	Status
Logical Issue, Inconsistency	Minor	projects/vinci-protocol/contracts/protocol/lendingpool/LendingPool.sol (1): 253~255; projects/vinci-protocol/contracts/protocol/tokenization/NToken.sol (1): 268~271	Acknowledged

Description

Function `depositAndLockNFT()` makes an external call to `INToken(nToken).lock()` function. However, the NFT lock is not implemented in nToken contract, and the the call may be reverted.

Contract LendingPool:

```
252        bool isFirstDeposit = INToken(nToken).mint(onBehalfOf, tokenIds[i], 1);
253        if(lockType != 0) {
254          INToken(nToken).lock(tokenIds[i], lockType);
255        }
```

Contract NToken:

```
268    function lock(uint256 tokenId, uint16 lockType) public virtual override
onlyLendingPool
269    {
270      revert('LV_NFT_LOCK_NOT_IMPLEMENTED');
271    }
```

Recommendation

Recommend to either remove the external call or implement the lock, to make sure that the contract operations won't be blocked.

Alleviation

[Vinci Team]: This is working as intended. For pools that support locking, we will use TimeLockableNToken as the NFT Token.

PRO-02 | Shadowing Built-In Symbol

Category	Severity	Location	Status
Coding Style	Informational	projects/vinci-protocol/contracts/protocol/libraries/logic/ValidationLogic.sol (1): 65; projects/vinci-protocol/contracts/protocol/tokenization/TimeLockableNToken.sol (1): 37	Acknowledged

Description

A user-defined component is shadowing a built-in symbol.

- In contract `ValidationLogic`, function `validateLockNFT()`, the variable `now` is shadowing a built-in symbol.

```
65   function validateLockNFT(DataTypes.NFTVaultData storage vault, uint40 now) external
view {
```

- In contract `TimeLockableNToken`, function `unlockedBalanceOfBatch()`, the variable `now` is shadowing a built-in symbol.

```
37     uint256 now = block.timestamp;
```

Recommendation

We recommend removing or renaming the declaration that shadows a built-in symbol.

VLC-01 | Array Length Not Checked Before Loop

Category	Severity	Location	Status
Volatile Code	Minor	projects/vinci-protocol/contracts/protocol/libraries/logic/ValidationLogic.sol (1): 114, 118, 274, 277	Acknowledged

Description

In function `validateWithdrawNFT()` and `validateNFTFlashloan()`, the code performed the check on length for `tokenIds` and `amount`, while `userBalance` is left out. The unaligned `userBalance` could cause the failure in the *for* loop.

Recommendation

Recommend to add a require check on the length of `userBalance` as well.

Alleviation

[Vinci Team]: Issue acknowledged. I won't make any changes for the current version.

All calls to these two functions are guaranteed to have the same length for userBalance and tokenIds.

WBP-01 | Locked Ether In Contract WalletBalanceProvider

Category	Severity	Location	Status
Language Specific	Minor	projects/vinci-protocol/contracts/misc/WalletBalanceProvider.sol (1): 36	Resolved

Description

The contract has a payable function that can be called by contract, but does not have a method to withdraw the ETH.

```
36    receive() external payable {
37      //only contracts can send ETH to the core
38      require(msg.sender.isContract(), '22');
39    }
```

Recommendation

We recommend removing the `payable` attribute or adding a withdraw logic or function.

Alleviation

[Vinci Team]: Issue acknowledged. Changes have been reflected in the commit hash 1843696:

https://github.com/VinciProtocol/vinci-protocol/commit/1843696b155352faa6285354b57abb5471515025

Optimizations

ID	Title	Category	Severity	Status
DRI-01	Unused Function `_getOverallBorrowRate()`	Gas Optimization	Optimization	Resolved
WEC-01	Variables That Could Be Declared As `constant`	Gas Optimization	Optimization	Acknowledged

DRI-01 | Unused Function `_getOverallBorrowRate()`

Category	Severity	Location	Status
Gas Optimization	Optimization	projects/vinci-protocol/contracts/protocol/lendingpool/DefaultReserveInterestRateStrategy.sol (1): 235	Resolved

Description

Internal function `_getOverallBorrowRate()` is not used in the contract.

Recommendation

We advise the client to remove it if there is no plan for further usage.

Alleviation

[Vinci Team]: Issue acknowledged. Changes have been reflected in the commit hash 26bc18ed829d8bda3f0824c2501b3cc99fc2ba2c

WEC-01 | Variables That Could Be Declared As `constant`

Category	Severity	Location	Status
Gas Optimization	Optimization	projects/vinci-protocol/contracts/dependencies/weth/WETH9.sol (1): 19, 20, 21	Acknowledged

Description

The linked variables could be declared as `constant` since these state variables are never modified.

Recommendation

We recommend to declare these variables as `constant`.

Appendix

Details on Formal Verification

Technical description

All Solidity smart contracts from the project that implement the ERC-20 interface are in scope of the analysis. Each such contract is compiled into a mathematical model which reflects all possible behaviors of the contract. All subsequent verification results are based on that model, which is designed specifically to be amenable to automated analysis by theorem provers and symbolic model checkers. Apart from representing all possible behaviors of the smart contract, the model also incorporates a verification harness that formalizes the initialization and interaction patterns for the contract. In particular, we use a verification harness that non-deterministically selects a public or external function and models its execution. The contract state is initialized non-deterministically (i.e. by arbitrary values) before invocation of the function. Hence, the mathematical model over-approximates the reachable state space of the contract throughout any actual deployment on chain. By doing so, all verification results carry over to the contract's behavior in arbitrary states after it has been deployed. Once the model is constructed, our analysis engine attempts to prove that all executions of the contract are subsumed by a set of pre-defined specifications which capture the desired and admissible behaviors of the smart contract. For the scope of this audit, we use 38 property specifications that cover the functionality of the functions as stated in Sec. Scope.

Assumptions and simplifications

The following assumptions and simplifications have been applied during formal verification:

- Gas consumption is not taken into account, i.e. we assume that executions do not terminate prematurely because they run out of gas.
- The contract's state variables are non-deterministically initialized before invocation of any of those functions. That ignores contract invariants and may lead to false positives. It is, however, a safe over-approximation.
- The verification engine reasons about unbounded integers. Machine arithmetic is modeled as operations on the congruence classes arising from the bit-width of the underlying numeric type. This ensures that over- and underflow characteristics are faithfully represented.

Formalism for property definitions

This section provides details on the 38 formal specifications that were in scope of the audit. All properties are expressed in linear temporal logic (LTL). In that context, we consider all invocations and returns from public and external functions as discrete time steps. Thus, our analysis reasons about the contract's state upon entering and leaving public and external functions.

Apart from the Boolean connectives and the modal operators "always" (written `[]`) and "eventually" (written `<>`), we use the following predicates to reason about the validity of atomic propositions. They are evaluated on the contract's state whenever a discrete time step occurs:

- `started(f, [cond])` Indicates an invocation of contract function `f` within a state satisfying formula `cond`.
- `willSucceed(f, [cond])` Indicates an invocation of contract function `f` within a state satisfying formula `cond` and considers only those executions that do not revert.
- `finished(f, [cond])` Indicates that execution returns from contract function `f` in a state satisfying formula `cond`. Here, formula `cond` may refer to the contract's state variables and to the value they had upon entering the function (using the `old` function).
- `reverted(f, [cond])` Indicates that execution of contract function `f` was interrupted by an exception in a contract state satisfying formula `cond`.

The verification performed in this audit is restricted to pre- and postconditions of procedure invocations. The used model consists of a harness that invokes a non-deterministically selected function of the contract's public and external interface. All formulas are analyzed w.r.t. the trace that corresponds to this function invocation.

Properties for ERC-20 function `transfer(to, amount)`

erc20-transfer-correct-amount

It is expected that non-reverting invocations of `transfer(recipient, amount)` that return `true` subtract the value in `amount` from the balance of the address `msg.sender` and add the same value to the balance entry of the `recipient` address.

```
[](willSucceed(transfer(to, value), to != msg.sender)
   ==> <> (finished(transfer(to, value),
                    return == true
                    ==> balance[msg.sender] == old(balance[msg.sender]) - value
                        && balance[to] == old(balance[to]) + value)))
```

erc20-transfer-correct-amount-self

It is expected that non-reverting invocations of `transfer(recipient, amount)` that return `true` and where the address in `recipient` equals the address of `msg.sender` (i.e. self-transfers) do not change the balance of address `msg.sender`

```
[](willSucceed(transfer(to, value), to == msg.sender)
   ==> <> (finished(transfer(to, value),
                    return == true
                    ==> balance[to] == old(balance[to]))))
```

Properties for ERC-20 function `transferFrom(from, to, amount)`

erc20-transferfrom-revert-from-zero

It is expected that calls of the form `transferFrom(from, dest, amount)` fail if the address value provided in the `from` in-parameter is the zero address.

```
[](started(transferFrom(from, to, value), from == 0)
   ==> <> (reverted(transferFrom) || finished(transferFrom, return == false)))
```

erc20-transferfrom-revert-to-zero

It is expected that calls of the form `transferFrom(from, dest, amount)` fail if the address value provided in the `dest` in-parameter is the zero address.

```
[](started(transferFrom(from, to, value), to == 0)
   ==> <> (reverted(transferFrom) || finished(transferFrom, return == false)))
```

erc20-transferfrom-fail-exceed-balance

Any call of the form `transferFrom(from, dest, amount)` with a value for `amount` that exceeds the balance of address `from` is expected to fail.

```
[](started(transferFrom(from, to, value), value > balance[from])
   ==> <> (reverted(transferFrom) || finished(transferFrom, return == false)))
```

erc20-transferfrom-correct-allowance

It is expected that non-reverting invocations of `transferFrom(from, to, amount)` that return `true` decrease the allowance of the address in `msg.sender` for the address in `from` by the value in `amount`. Two special cases are taken into account:

1. An allowance that equals `type(uint256).max` is treated as an exception and interpreted as an unlimited allowance that does not need to be reduced in order for this check to pass.
2. If the owner of the tokens that are transferred invokes `transferFrom` (i.e. when the address in `msg.sender` equals the address in `from`) we do not require an update of the allowance.

```
[](willSucceed(transferFrom(from, to, value))
   ==> <> finished(transferFrom(from, to, value),
                   return == true
                   ==> ((allowances[from][msg.sender] == old(allowances[from]
[msg.sender]) - value)
                        || (allowances[from][msg.sender] == old(allowances[from]
[msg.sender])
                            && (from == msg.sender
                                || old(allowances[from][msg.sender]) ==
type(uint256).max)))))
```

Finding Categories

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.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

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.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

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.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

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

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.

About

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.