CERTIK

Preliminary Comments

Sovryn - Smart Contracts

Oct 11th, 2021

Table of Contents

Summary

This report has been prepared for Sovryn to discover issues and vulnerabilities in the source code of the Sovryn - Smart Contracts 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	Sovryn - Smart Contracts
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/DistributedCollective/Sovryn-smart-contracts
Commit	24f331963c9d54d4d5a5aa58fd51f00c7cbd7ee6

Audit Summary

Delivery Date	Oct 11, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	26	26	0	0	0	0
Medium	6	6	0	0	0	0
Minor	19	19	0	0	0	0
Informational	55	55	0	0	0	0
Discussion	2	2	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
FLS	connectors/loantoken/interfaces/FeedsLike.sol	69162fc20ed2a33904687682efa6d91271218affad56d91ee9dddcc601e25ce1
PLS	connectors/loantoken/interfaces/ProtocolLike.sol	eb7b1529d585e08939a8fc8801dd428066721e1b7e70c2f1025cb27a2c1995d5
PSL	connectors/loantoken/interfaces/ProtocolSettingsLike.sol	52c491d07fc6ec105eff18358715a777334f1bc740669e292c3e951f458d975f
ATS	connectors/loantoken/AdvancedToken.sol	0afb07be23a7266d773dac65ab9589a184a286ff92ebd1d71840e12c434987b1
ATD	connectors/loantoken/AdvancedTokenStorage.sol	4caab694fe6c7906cc08b4d23c4535c39d121b36b765ceea48e04ba0e38a9375
LTS	connectors/loantoken/LoanToken.sol	644a121e34cd6d9e5c0692639cd28c838f59565fdf63377acf7621ddfe4024a1
LTB	connectors/loantoken/LoanTokenBase.sol	c20b3c55c75d29ffd97be63ab24a19abdea9b1af39a0420336ac4a491a366336
LTL	connectors/loantoken/LoanTokenLogicLM.sol	1e8449a9f38d7b7ab1abf1b09c55559a26362367c8084ea82ea9413805a650bd
LTD	connectors/loantoken/LoanTokenLogicStandard.sol	b6553a3682c8455e53d2b77f370817e91f55c0a9ab1637dbc200c9ecccd1d6a5
LTW	connectors/loantoken/LoanTokenLogicWrbtc.sol	54dcd444d1dd8df2cc3003ffdf2be8f41f81fc97a9a816fe379979497321b410
LTA	connectors/loantoken/LoanTokenSettingsLowerAdmin.sol	d205b5e7118b5c2c91a1a2c6ff289a8f2d385c8d9e8d2fc8638f3d6293f23055
PSD	connectors/loantoken/Pausable.sol	75a6107ea46c31628393763930fdf913606e43ef2727601af5aebdc1ce38095d
LIS	core/objects/LenderInterestStruct.sol	6eb315487042aa2d42a4d854a57028fec700b04d4fec6e848b45ef5b65899cf7
LID	core/objects/LoanInterestStruct.sol	df4287d4ce977953db68b625dd6bb9fefdefc55e708b53d5f91b046776ed66b0
LPS	core/objects/LoanParamsStruct.sol	82d8386f22c159d69e945ee821544e189dfd1a44da758a62fee100ccb3c834f7

ID	File	SHA256 Checksum
LSS	core/objects/LoanStruct.sol	3d4c92d3c1d2b1949c2c6632c33971ae5d1df5183cb743d434c99e0e643dd7f6
OSS	core/objects/OrderStruct.sol	61421e1f319638c6de89a9c75145fac4a248d8dba191fcd9f495955f855bd0eb
OSD	core/Objects.sol	ac2e27ac3cb822fe25e131b8e0b620a61becb41da87513e77759c838fdc3b3cd
PSC	core/Protocol.sol	72b2627047b603b5889efdb25672a41ed83fe006a70f91ed90f8696499f3fef9
SSD	core/State.sol	d839cec56fedfab8f8438d118249ebea7f483d648f4f3ce1651e5a4deb0b025c
ESD	escrow/Escrow.sol	b876cdf91d22f2af91ae3fd365537987872626 44e2ac03afe4a8e9378d53f32d
ERS	escrow/EscrowReward.sol	226a7f59c8d5991fc45957cc8ad05590b6d51cccc08e2a2a17005f7a4d58e9d6
AES	events/AffiliatesEvents.sol	485260e38f8f6d0a5a66e16d0ff37fe3841c9102e83b4f8ca7bce667d2b44d86
FES	events/FeesEvents.sol	753ddac4eb8b347ddee4cf23aad38e326f17e7b99155b5801462fe1de46be149
LCE	events/LoanClosingsEvents.sol	ed1fcc48e4693e2cba54657c208e579e7e40df3e7529ad682f72bf74545aae82
LME	events/LoanMaintenanceEvents.sol	ab0c3ea4e9ff83977fb8f5c16b2a13d124a0ac001930a08a730a63ee0f17e235
LOE	events/LoanOpeningsEvents.sol	cec3010da17c5c6a69845022ea18f086a2b6688e02bfe8e194fd9209c7f1edb1
LSE	events/LoanSettingsEvents.sol	61891e818c5656212a1488bb86c842ccca2ad3ddfcb0ccd8251d8fccafb7b238
MCE	events/ModulesCommonEvents.sol	30cc6ea96be9bcef88d59567557f1ded9a63bf56c8d6c823af03225547a489af
PSE	events/ProtocolSettingsEvents.sol	af2403631de3afe131e44bcca7e64220b32e99428916cda7f51a515e7003c4dd
SES	events/SwapsEvents.sol	11cb8e00d9a3b47217ba79a19422b32f4688bd0cc75e67b25ae4c73ba469546f
ILM	farm/ILiquidityMining.sol	97e993f2705f9b1fb59d7d31fe715c93b0551650d90b67cff6ec98f0db372777

ID	File	SHA256 Checksum
LMS	farm/LiquidityMining.sol	5fcdc611c1363d28db742a520465c378f6dcd6467f760552a01087df3a2ed773
LMC	farm/LiquidityMiningConfigToken.sol	0362dfdc92aee8586e3754b3f9e0904c5f88ccf9d8445116109407cf59f9b701
LMP	farm/LiquidityMiningProxy.sol	a31863a7fd41c3c46c1d027aa8cb1739878e59c4e189cc183413f38872bffa28
LMD	farm/LiquidityMiningStorage.sol	4f7c48e1558c8dde9ea5ce5f1795c30b15ae0453ba6586e3cc08626ddda784a0
PFL	feeds/testnet/PriceFeedsLocal.sol	cc1662e4c7cf30d64d08d2342b4f01ef735b96133bc99fed0e586b09c54db37c
PFM	feeds/testnet/PriceFeedsMoC.sol	4a5d72a2175519aba415750c7a57dbe44de54f747343b0ac1cfc74d1fff831ba
BPP	feeds/BProPriceFeed.sol	668ac568bca30b60216b626b9d15ad73276211281fc80d470e4d98890896832b
IMC	feeds/IMoCState.sol	fc5c985911206b2084534e683949129916a0d2c87d55ddd8f2dccb90a0a718d8f
IPF	feeds/IPriceFeeds.sol	78ac11b9f8dbddae05d57d49281e3fb981011bb7260cf88bc37e0dd0e3bdfff2
IRS	feeds/IRSKOracle.sol	233f2ae433ff503e2612f08d66f27e46f9c7bf296baffad594c513b60102eb34
IVP	feeds/IV1PoolOracle.sol	41bb7c6594bcabcd3730485c3351065f591e3c2314c4b883e7203a0c51295f4a
PFR	feeds/PriceFeedRSKOracle.sol	6235141c26978b644a43ed5c4d0a1e4c0a0ed21be9bd6e4b0e3f46feadca7453
PFV	feeds/PriceFeedV1PoolOracle.sol	3b4f9724150d22b16f0e161f0dff6e4403d15ac807130050cae109b7c84a5192
PFS	feeds/PriceFeeds.sol	1571be122fee70c7f76b2ccd1530fa072c98ed398adafd35ad649ce0976a0b03
PFC	feeds/PriceFeedsConstants.sol	f6cd49902887ca177e28393f5af63ed7e1ba487134e5b8541492600110029d0f
USD	feeds/USDTPriceFeed.sol	bf6bad2da211c518810025646e225c9bc0d640ee7dc94ba0dbc67bbf5af52c9c
CSS	governance/Staking/Checkpoints.sol	d1680c99f497e35c3d8629beff3fba6bed44b44e417011a8bdfaba3b95d46d69

ID	File	SHA256 Checksum
ISS	governance/Staking/IStaking.sol	6e2358901581 3aba92568a196d0d9ce75613009b39ed87410ec98a669ef0e4f
SMS	governance/Staking/SafeMath96.sol	99d6a08d3f21288ba93d3c96b527e09d37543188c7035562246332c973fe393e
SSS	governance/Staking/Staking.sol	1152fb9357b3d4ee45fc5df6d0a819d9260e844a227a0eb5b671a9816e25de60
SPS	governance/Staking/StakingProxy.sol	2923142990472ad689665d73dfb906c1852ee4a78a0d002291eec586697de108
STA	governance/Staking/StakingStorage.sol	cb1d859ff79cf5d75b958ac49b4f00b810a4f7115bdf9aa3b022d042ae86d001
WSS	governance/Staking/WeightedStaking.sol	85d2e85592134c019f6cb3f242c29f5375ea446840f506350f89470ca524fc95
SRR	governance/StakingRewards/StakingRewards.sol	4f02e50026986312a6a2d846a3e21c4126796350ed55bce8532ee9432c3f87ec
SRP	governance/StakingRewards/StakingRewardsProxy.sol	3019ead2b158d6ec49466ad350f9ec7c90a81b23973811efd8b8d703cc3bee21
SRD	governance/StakingRewards/StakingRewardsStorage.sol	f4177189f434921a20be025f3a9d8e15a91732fa383b7cc99195e2a8dc2cf5d1
DFV	governance/Vesting/DevelopmentFund.sol	818c5730e05e4fffd39690dc185a7e46ffbb2889122f9992f38ad93dd94050d6
ITV	governance/Vesting/ITeamVesting.sol	b59510e3c9aa661cb35772f94d16153d8f018a60456f4c8c384ad54382eb622d
IVV	governance/Vesting/IVesting.sol	50fbd299d61e1a931b2bbc7405b51f341eab8770739b4f5f8d4815a0e5af9d28
IVF	governance/Vesting/IVestingFactory.sol	4022128df0c368d73ca95b340c51a54dfd920ab441fc0ba20c0beb77b265071a
IVR	governance/Vesting/IVestingRegistry.sol	259a9faeec2866b090ca20e76b284fa7360948ca244d7a1bba0ebea56d9afb0a
OIC	governance/Vesting/OriginInvestorsClaim.sol	cc7a9411687f937824f21a8d8b22bd5067e9a112b33bbf00f70a0db4f0a9bc6b
OVC	governance/Vesting/OrigingVestingCreator.sol	5cd086fb5dd8b1c7aa6278fc1d82fd96898599a61b1c94805d2cf6305cb48556

ID	File	SHA256 Checksum
SVR	governance/Vesting/SVR.sol	160c0ab1c6a83ac79188cb25336b266213980b0b518a692c0949762ba526d91b
TVV	governance/Vesting/TeamVesting.sol	68fe73b49dc3632b85178e151aa5fa89e7d5ddd5e675ee040802a82c472c9c6f
TSV	governance/Vesting/TokenSender.sol	d20730c34e6b0b5c251ac156f0ca08437a28e91df8301769d0ef70b58974fd0b
VVS	governance/Vesting/Vesting.sol	fa3a2e0a6c6c8858d5bb394388d4dcc9926c472b58653a052d6d5bbf1d20e65c
VCV	governance/Vesting/VestingCreator.sol	7c6b3b66ae7533606703060ea637651452e0c24c525cce02d221579d8e16a21b
VFV	governance/Vesting/VestingFactory.sol	f42b645520a4a3708a10d8503f1072e8e9d0bcaed66c09fcde4ac188c9ceded5
VLV	governance/Vesting/VestingLogic.sol	af6da24e3c532895bd680f1fe02d030cf0c4f1f15020e35078efcbecf9110498
VRV	governance/Vesting/VestingRegistry.sol	1f3bc7499ee6f974dbbc37ea6bc797c039e725c0ace38cb391e6a955d8f51edb
VRS	governance/Vesting/VestingRegistry2.sol	af06e7c12b0376428ccaf02ca398139ab4a55cbf6cd36fddcf41c20bdcbafa7d
VRD	governance/Vesting/VestingRegistry3.sol	2fa39ab5ca13c4fefc7d413e35b2494adcaa00552fbefbe5e3a2a80c32653464
VRL	governance/Vesting/VestingRegistryLogic.sol	9d5d4f1b08a370770a70aa792017559504eb8d6138bb6910b70f263e4077d304
VRP	governance/Vesting/VestingRegistryProxy.sol	a04ce3f37a3ac99ccd3b56cfa4fee98641b9aa1747c5a8613341b3f2a621b663
VRC	governance/Vesting/VestingRegistryStorage.sol	d5a395cefbb8ca7293963d2590a5ac8a8e7182081da55bd85b37758ca8e93577
VSV	governance/Vesting/VestingStorage.sol	927c0e2d4a4da9a4a0d11f78b4d423eb1e00709018489e81087d2cce318d771c
ARS	governance/ApprovalReceiver.sol	73c2ef000a1d7011c45ce8f64e97ee7e2399cd22b0119bf29bc0bcab60a1f5cd
EDS	governance/ErrorDecoder.sol	9f38e0981c4dfdeda21e5f0f4e57e656052c56d57d88a779b8008ef45ea72b11
FSP	governance/FeeSharingProxy.sol	41d063d027a20391d8f5af244bb743500dd016a9704e3e8896648e13d414164d

ID	File	SHA256 Checksum
GAS	governance/GovernorAlpha.sol	975b0f5745a36fb0a0d48f4cd6b38a939f1c7ba280e43331909fc3ef16404d81
GVS	governance/GovernorVault.sol	065cbb5a2a2bef66db4339d1249617009e31000ce384bda881f118c2abeaffdd
IFS	governance/IFeeSharingProxy.sol	7530111d6e5608a124ff04a2456cf780a7832b8a0a904dad192ee812307e3d86
TSD	governance/Timelock.sol	2b11a4b8d969f4d7c000c021ec7008400f43d905ed4c099ceda274c0e8044431
ICS	interfaces/IChai.sol	89b0199091d7418720a6544cb40544a3fbdfb64c843e42956c51899d27fb6e9c
IER	interfaces/IERC20.sol	f2a0514295264a1859a265e7faa0c2ef6207ca8216f465f02e1e3a0fac7b1663
ILP	interfaces/ILoanPool.sol	6b5661b22f999ecced4b9c8a559a961529ca9afb39d902b6f662e98c7c84b5c2
ISD	interfaces/ISovryn.sol	4e8dac38aa2421fc2a9fa88ad791c527cbf3f51777f3cef0a80d68230f435440
IWS	interfaces/IWrbtc.sol	ae9b3de38a98385b0f72dc990ef86f28f50c2f7729309f621d299e0f22ec3db8
IWE	interfaces/IWrbtcERC20.sol	e8f93aad947048d641e1791ad1caa0f58d9861d7e0c3bfc6fd2d4166ed8d7126
ILS	locked/ILockedSOV.sol	58bf5ab2a7464e7dc48b57eddde3a869c04d056e1c223e8acfcf26d88dceeef1
LSO	locked/LockedSOV.sol	80ec65fa2bbd977c6c9930c67a8b90b94a5dad379647c57db3a91b446deca871
EAS	mixins/EnumerableAddressSet.sol	4710cb72d9291929744f0d7c16f76c8fd9412f6044ed57ebc0c5723f03194075
EBS	mixins/EnumerableBytes32Set.sol	f19210b18463f6e1bf5364118a5b4c95934100075b324b6feac3d83016dcfed0
FHS	mixins/FeesHelper.sol	5ce6e7dcb182352785724c32aa30d2e97309e47366706f0d9c37bacaee4668b1
IUS	mixins/InterestUser.sol	99cb14e6afb8af46659bc1a1b4c672c0239f431c9f29d7873876996b67d5c10e
LHS	mixins/LiquidationHelper.sol	9221c85f70841c4b0a9edd27d17223ccbc07439accbbb3236f2325620edc85ce

ID	File	SHA256 Checksum
PTU	mixins/ProtocolTokenUser.sol	8b8265c4cf688363c168434913796517b1e7b92267612427ddf2764e12eb443a
RHS	mixins/RewardHelper.sol	dd1ded62f9e71e34c05ae0ee6158f46061abb37b6bd20fc021b4f7c0c5594f5d
VCS	mixins/VaultController.sol	aac5e2e9070073bc689be12b380502bc4dd4d42495f6b093f5b09a067a4a63ee
PAI	modules/interfaces/ProtocolAffiliatesInterface.sol	9efc6721a3bc52cd6293aeaf168a0b9948a0b295676a355c1f53267afc760611
ASD	modules/Affiliates.sol	f319afcfa4f628860ff5775ef416a455c53d816f30345581e1304643cc395585
LCB	modules/LoanClosingsBase.sol	97cecfe2b5d6cdc0ae58275171a6db08c3bee750db05a866cb7efb69ef323b73
LCW	modules/LoanClosingsWith.sol	5cf8d5d892d3d0695b3c57cfa6921b6f9cd9d8970a7bc160444d58ebc68a10f7
LSD	modules/LoanMaintenance.sol	c1521565132122b88c2d761cdd5262243a6dc3605a4313331249ffdd63708228
LOS	modules/LoanOpenings.sol	169c86233117f6cbb7a0c15ed5383495137d53a9f0d5e7fbdd8670224fcfb2f3
LSC	modules/LoanSettings.sol	9c1c36eb9d70996be73857f7a719a1aadafa5ccc33ed7fb7076f0a0f6d9d88b3
MCF	modules/ModuleCommonFunctionalities.sol	90837e34a7ff27c4d9d4d2024b5f936f12f26f19c255913eb58c5c409a8cb8e6
PSS	modules/ProtocolSettings.sol	5a66fabfe5e393b07760eb890601d1e5d4f7c99dee24c6a40243dea6234d70f4
SED	modules/SwapsExternal.sol	bbbdaec98596d7d42016a814e959f808eed7fab904b2af21e32dc2eff081f7c3
MSK	multisig/MultiSigKeyHolders.sol	ade7a4902390e0c3fa6431e980901609d03c8379a067c90bb4f2104ef4e532b6
MSW	multisig/MultiSigWallet.sol	e8bba6c3dfc919638f10dce617f180488ba2e84aa1aa226f7bfadd411e959726
PDC	proxy/Proxy.sol	8a392a994f040f857ffb1e76e2ca2e4b506e24a36645c087135180f691f3a437
UPS	proxy/UpgradableProxy.sol	2321fbda1191801da556d6fb21929167ed0a9bd60a875ca3a482969a8402d4d6

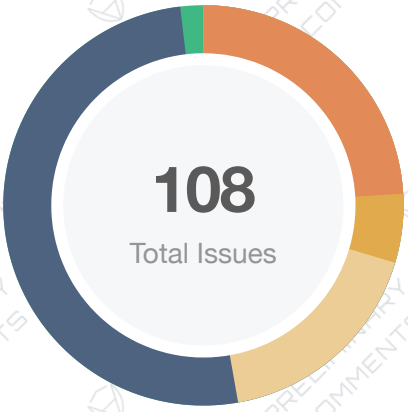
ID	File	SHA256 Checksum
RSA	rsk/RSKAddrValidator.sol	94d6bd264ddd8c78ddb215a9f5dba128167ace5a35fd8d5982c66f31e3962274
ICR	swaps/connectors/interfaces/IContractRegistry.sol	470f364a6987bf53ac2335b402feba974e393ae0010f33a2346709206b080d83
ISN	swaps/connectors/interfaces/ISovrynSwapNetwork.sol	73271a239205c83c55ed6c680c0466c14fbf574d877e6dd8621d6201c6e1c2e7
SIL	swaps/connectors/testnet/SwapsImplLocal.sol	e2b284647de75d09ef7004e76bda037ff9b994142a00120eefb7af1aff13d70f
SIS	swaps/connectors/SwapsImplSovrynSwap.sol	faa3b8b6da4d32a5e1a3f02248aeccbac4fc69a2c22a3fd32501fd58a8eb1090
ISI	swaps/ISwapsImpl.sol	049d188aa2d977b447ce082cfe66857a1a43bdef1fe30a477ca210fd3d62c65a
SUS	swaps/SwapsUser.sol	7e6ae0110015f82aaa5750912d5e883d8f3c873bbe6596a2396f5b21ce75264c
IAA	token/IApproveAndCall.sol	2cd1c7c74e307d60e57bb9901b33c7b8d68c74ffa6c4b4a21a7694a246a98c36
SOV	token/SOV.sol	0681486b7c1b2b6f60403f476d24a5b7530d89a80abb882bbc6cca4fbe0fc198
ARD	utils/AdminRole.sol	0a399f0836b50e489dea7f4c65945c717e0af612e338f99ed4f87c73a6ad1853

Executive Summary

The audited codebase comprises the following features implemented in smart contracts.

- Borrowing, lending, and margin trading of assets that allow lenders to earn interest by lending the assets and getting iToken, and borrower pay interests for the duration of the loan that can be rolled over. The borrow positions are eligible for liquidation if the margin falls below the maintenance margin.
- Checkpoints-based staking of SOV tokens that keeps track of stakes based on users and timestamps. These checkpoints based staking allows determining stakes of users against a particular date that serves as voting weight in the governance contract to determine the eligibility of users to create and vote proposals.
- Vesting contracts that allow vestings of staking rewards and team allocation.
- An multisig smart contract implementation that allows executions of transactions having reached the required quorum.
- Oracle price feeds contracts that allow retrieving of conversion rates and precision between the whitelisted assets. These contracts also provide retrieval of margin, available drawdown, and liquidation status of loan positions.
- Swap contracts that perform asset swaps for loans and trades, and paying trading fees.
- Liquidity mining contracts that allow staking of whitelisted pool tokens by the users and are rewarded with SOV tokens that are sent to `LockedSOV` contract where the `unlockedImmediatelyPercent` portion of the reward is immediately claimable while the rest is vested.

Findings



Critical	0 (0.00%)
Major	26 (24.07%)
Medium	6 (5.56%)
Minor	19 (17.59%)
Informational	55 (50.93%)
Discussion	2 (1.85%)

ID	Title	Category	Severity	Status
ARS-01	Redundant Variable Initialization	Coding Style	Informational	Pending
ASD-01	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
ASD-02	Explicitly Returning Local Variable	Gas Optimization	Informational	Pending
ASD-03	Explicitly Returning Local Variable	Gas Optimization	Informational	Pending
BPP-01	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
DFV-01	Missing Validation for The Past `lastReleaseTime`	Logical Issue	Minor	Pending
DFV-02	Ineffectual Security of Token Owner Lock/Unlock Mechanism	Logical Issue	Discussion	Pending
DFV-03	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
DFV-04	Function Visibility Optimization	Gas Optimization	Minor	Pending
DFV-05	Redundant `SafeMath` Utilizations	Gas Optimization	Informational	Pending
ESD-01	Redundant Statements	Volatile Code	Informational	Pending

ID	Title	Category	Severity	Status
ESD-02	Storage Variables Are Not Packed	Gas Optimization	Informational	Pending
ESD-03	Redundant Statements	Volatile Code	Informational	Pending
ESD-04	Inefficient Storage Read	Gas Optimization	Informational	Pending
ESD-05	Inefficient Storage Read	Gas Optimization	Informational	Pending
FSP-01	Missing Input Validation	Logical Issue	Minor	Pending
FSP-02	Unneeded Packing of Local Variable	Gas Optimization	Informational	Pending
FSP-03	Potential Volatile Implementation	Logical Issue	Discussion	Pending
FSP-04	Requisite Value of ERC-20 `transferFrom()` / `transfer()` Call	Logical Issue	Minor	Pending
FSP-05	Inefficient Storage Read	Gas Optimization	Informational	Pending
GAS-01	Division Before Multiplication	Mathematical Operations	Minor	Pending
GAS-02	Potential Logic Flaw in `state()`	Logical Issue	Medium	Pending
GAS-03	Function Visibility Optimization	Gas Optimization	Informational	Pending
GAS-04	Comparison with `boolean` Literal	Gas Optimization	Informational	Pending
GAS-05	Inefficient Storage Read	Gas Optimization	Informational	Pending
GVS-01	Usage of `transfer()` for Sending Ether	Volatile Code	Minor	Pending
LCB-01	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
LCB-02	Dangerous Usage of `tx.origin`	Volatile Code	Minor	Pending

ID	Title	Category	Severity	Status
LCW-01	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
LMS-01	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
LMS-02	Redundant Statements	Volatile Code	Informational	Pending
LMS-03	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
LOS-01	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
LSC-01	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
LSD-01	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
LSO-01	`AdminAdded` Event Is Not Emitted	Volatile Code	Informational	Pending
LSO-02	Storage Variables Are Not Packed	Gas Optimization	Informational	Pending
LSO-03	Return Variable Utilization	Gas Optimization	Informational	Pending
LSO-04	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
LTA-01	Function Visibility Optimization	Gas Optimization	Informational	Pending
LTA-02	Missing Input Validation	Volatile Code	Minor	Pending
LTA-03	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
LTB-01	Inefficient Storage Layout	Gas Optimization	Informational	Pending
LTD-01	Documentation Discrepancy	Inconsistency	Informational	Pending
LTD-02	Missing Input Validation	Volatile Code	Minor	Pending

ID	Title	Category	Severity	Status
LTD-03	Inefficient Storage Read	Gas Optimization	Informational	Pending
LTD-04	Missing Error Messages	Coding Style	Informational	Pending
LTD-05	Return Variable Utilization	Gas Optimization	Informational	Pending
LTD-06	Inefficient Storage Read	Gas Optimization	Informational	Pending
LTD-07	Potential Allowance Overwritten	Logical Issue	Minor	Pending
LTD-08	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
LTS-01	Contract's State Can Be Re-initialized	**Centralization / Privilege**	**Major**	Pending
LTS-02	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
MSK-01	Function Visibility Optimization	Gas Optimization	Informational	Pending
MSK-02	Ethereum Addresses Count Can Exceed `MAX_OWNER_COUNT`	Logical Issue	Minor	Pending
MSK-03	Bitcoin Addresses Count Can Exceed `MAX_OWNER_COUNT`	Logical Issue	Minor	Pending
MSW-01	Inefficient Storage Struct Layout	Gas Optimization	Informational	Pending
MSW-02	Missing Error Messages	Coding Style	Informational	Pending
MSW-03	Inefficient Storage Read	Gas Optimization	Informational	Pending
OIC-01	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
OIC-02	Centralization Risk	**Centralization / Privilege**	**Major**	Pending

ID	Title	Category	Severity	Status
PFR-01	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
PFS-01	Redundant Variable Initialization	Gas Optimization	Informational	Pending
PFS-02	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
PSS-01	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
PSS-02	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
PTU-01	Possibility of Protocol Tokens Being Locked	Logical Issue	Minor	Pending
SED-01	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
SOV-01	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
SOV-02	Function Visibility Optimization	Gas Optimization	Informational	Pending
SRP-01	Inheritance Order Does Not Allow Expanding of `StakingRewardsStorage` Contract With Additional Storage Structures	Logical Issue	Medium	Pending
SRR-01	Inheritance Order Does Not Allow Expanding of `StakingRewardsStorage` Contract With Additional Storage Structures	Logical Issue	Medium	Pending
SRR-02	Inefficient Storage Read	Gas Optimization	Informational	Pending
SRR-03	Inaccurate Validity Check	Logical Issue	Medium	Pending
SSD-01	Potential Volatile Implementation	Volatile Code	Minor	Pending
SSS-01	Missing Sanity Validation	Logical Issue	Minor	Pending
SSS-02	Lack of Error Message	Coding Style	Informational	Pending

ID	Title	Category	Severity	Status
SSS-03	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
SSS-04	Redundant Variable Initialization	Coding Style	Informational	Pending
SSS-05	Inefficient Storage Read	Gas Optimization	Informational	Pending
STA-01	Visibility Specifiers Missing	Language Specific	Informational	Pending
STA-02	Redundant Variable Initialization	Gas Optimization	Informational	Pending
SVR-01	Lack of Error Message	Coding Style	Informational	Pending
SVR-02	Visibility Specifiers Missing	Language Specific	Informational	Pending
TSD-01	Missing Input Validation	Volatile Code	Minor	Pending
TSV-01	Function Visibility Optimization	Gas Optimization	Informational	Pending
TSV-02	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
VCV-01	Inefficient Storage Struct Layout	Gas Optimization	Informational	Pending
VCV-02	Missing Input Validation	Volatile Code	Minor	Pending
VCV-03	Comparison with `boolean` Literal	Gas Optimization	Informational	Pending
VCV-04	Function Return Value Ignored	Volatile Code	Informational	Pending
VCV-05	Potential Stuck Vesting Process	Volatile Code	Medium	Pending
VCV-06	Inefficient Usage of Local Storage Variable	Gas Optimization	Informational	Pending
VCV-07	Inefficient Storage Read	Gas Optimization	Informational	Pending
VLV-01	Lack of Error Message	Coding Style	Informational	Pending

ID	Title	Category	Severity	Status
VLV-02	Inefficient Storage Read	Gas Optimization	Informational	Pending
VRL-01	Inheritance Order Does Not Allow Expanding of `VestingRegistryStorage` Contract With Additional Storage Structures	Logical Issue	Medium	Pending
VRL-02	Redundant `modifier` Usage	Coding Style	Informational	Pending
VRL-03	Return Variable Utilization	Gas Optimization	Informational	Pending
VRS-01	Usage of `transfer()` for Sending Ether	Volatile Code	Minor	Pending
VRS-02	Redundant Variable Initialization	Coding Style	Informational	Pending
VRV-01	Usage of `transfer()` for Sending Ether	Volatile Code	Minor	Pending
VRV-02	Redundant Variable Initialization	Coding Style	Informational	Pending
VRV-03	Inefficient Storage Read	Gas Optimization	Informational	Pending
VRV-04	Explicitly Returning Local Variable	Gas Optimization	Informational	Pending
VRV-05	Inefficient Storage Read	Gas Optimization	Informational	Pending
VRV-06	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
WSS-01	Inefficient Storage Read	Gas Optimization	Informational	Pending

ARS-01 | Redundant Variable Initialization

Category	Severity	Location	Status
Coding Style	Informational	governance/ApprovalReceiver.sol: 31	Pending

Description

All variable types within Solidity are initialized to their default ""empty"" value, which is usually their zeroed out representation.

Particularly:

- `uint` / `int`: All `uint` and `int` variable types are initialized at `0`
- `address`: All `address` types are initialized to `address(0)`
- `byte`: All `byte` types are initialized to their `byte(0)` representation
- `bool`: All `bool` types are initialized to `false`
- `ContractType`: All contract types (i.e. for a given `contract ERC20 {}` its contract type is `ERC20`) are initialized to their zeroed out address (i.e. for a given `contract ERC20 {}` its default value is `ERC20(address(0))`)
- `struct`: All `struct` types are initialized with all their members zeroed out according to this table

Recommendation

We advise that the linked initialization statements are removed from the codebase to increase legibility.

ASD-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	modules/Affiliates.sol: 48	Pending

Description

In the contract `Affiliates`, the role `owner` has the ability to set the target address for external functions of the aforementioned contract.

Any compromise to the `owner` account allows the hacker to set the target address of its choice and insert malicious code.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

ASD-02 | Explicitly Returning Local Variable

Category	Severity	Location	Status
Gas Optimization	Informational	modules/Affiliates.sol: 307, 390	Pending

Description

The `return` statements on the aforementioned lines explicitly return the named return variables of the function. As the named returned variables of a function are implicitly return, so the explicit use of `return` statement in such cases is redundant.

Recommendation

We advise to remove the explicit `return` statements on the aforementioned lines.

ASD-03 | Explicitly Returning Local Variable

Category	Severity	Location	Status
Gas Optimization	Informational	modules/Affiliates.sol: 195	Pending

Description

The function on the aforementioned line explicitly returns local variable which increases overall cost of gas.

Recommendation

Since named return variables can be declared in the signature of a function, consider refactoring to remove the local variable declaration and explicit return statement in order to reduce the overall cost of gas.

BPP-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	feeds/BProPriceFeed.sol: 53	Pending

Description

In the contract `BProPriceFeed`, the role `owner` has the authority to change the sensitive state variable `mocStateAddress`. This introduces centralization risk as this contract address is used to fetch the `bPro/USD` price on L35 and a malicious contract can return wrong result.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

DFV-01 | Missing Validation for The Past `lastReleaseTime`

Category	Severity	Location	Status
Logical Issue	Minor	governance/Vesting/DevelopmentFund.sol: 148, 252	Pending

Description

The code on aforementioned lines set `lastReleaseTime` representing the starting time for funds release yet it does not validate the parameter to have already passed, which can result in the funds immediately claimable if the `lastReleaseTime` is set in the distant past.

Recommendation

We advise to introduce a check ensuring the parameter `lastReleaseTime` is greater-than or equal-to `block.timestamp`.

DFV-02 | Ineffectual Security of Token Owner Lock/Unlock Mechanism

Category	Severity	Location	Status
Logical Issue	Discussion	governance/Vesting/DevelopmentFund.sol: 186~218	Pending

Description

The comment on `L196` states that a locked owner should only be approved by an unlocked owner and the function `approveLockedTokenOwner` only allows unlocked owner to approve a new locked owner. This flow can be circumvented by the locked owner by calling the function `updateUnlockedTokenOwner` providing their own address, possessing control of the unlocked owner and then calling the `approveLockedTokenOwner` function.

Recommendation

We advise to revisit this functionality and introduce the code flow that cannot be circumvented.

DFV-03 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	governance/Vesting/DevelopmentFund.sol: 286, 347	Pending

Description

In the contract `DevelopmentFund`, the `lockedOwner` and `unlockedOwner` accounts have the authority to withdraw `SOV` funds of the contract.

Any compromise to these accounts may allow the hacker to any the contract.

Recommendation

We advise the client to carefully manage the aforementioned accounts' private keys to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

DFV-04 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Minor	governance/Vesting/DevelopmentFund.sol: 241	Pending

Description

The following functions are declared as `public`, contain array function arguments, and are not invoked in any of the contracts contained within the project's scope. The functions that are never called internally within the contract should have external visibility.

Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

DFV-05 | Redundant `SafeMath` Utilizations

Category	Severity	Location	Status
Gas Optimization	Informational	governance/Vesting/DevelopmentFund.sol: 267, 271	Pending

Description

The linked mathematical statements can be safely conducted using their "raw" counterparts as they basically offset the already-safely-conducted mathematical operations in the preceding `if` block.

Recommendation

We advise the team to remove redundant code.

ESD-01 | Redundant Statements

Category	Severity	Location	Status
Volatile Code	Informational	escrow/Escrow.sol: 39, 54	Pending

Description

The `Expired` option of the `Status` enum and the event `EscrowFundExpired` are not used in the contract.

Recommendation

We advise to remove the unused code from the contract to increase code legibility.

ESD-02 | Storage Variables Are Not Packed

Category	Severity	Location	Status
Gas Optimization	Informational	escrow/Escrow.sol: 40	Pending

Description

The enum type `Status` variable on the aforementioned line occupies a complete storage slot of the contract. As the enum `Status` is represented by `uint8` under the hood, we can tight pack the enum variable and address type variable on L28 in a single slot by placing them next to each other.

Recommendation

We advise to place enum and address type variables on the aforementioned lines next to each other to save gas cost associated with additional storage slot.

```
address public multisig;
Status public status;
```

ESD-03 | Redundant Statements

Category	Severity	Location	Status
Volatile Code	Informational	escrow/Escrow.sol: 138	Pending

Description

The aforementioned line redundantly assignes default value of `Status.Deployed` to `Status` type storage variable.

Recommendation

We advise to remove the redundant assignment on the aforementioned line.

ESD-04 | Inefficient Storage Read

Category	Severity	Location	Status
Gas Optimization	Informational	escrow/Escrow.sol: 196~197, 205	Pending

Description

The code on aforementioned lines read `totalDeposit` from contract's storage which causes increased gas cost as the code can be rectified to limit the storage read to only once.

Recommendation

We advise to rectify the code on the aforementioned lines by introducing a local variable to limit the storage read of `totalDeposit` to only once to save gas cost associated with the extra storage read operation.

ESD-05 | Inefficient Storage Read

Category	Severity	Location	Status
Gas Optimization	Informational	escrow/Escrow.sol: 196~197	Pending

Description

The code on aforementioned lines read `depositLimit` from contract's storage twice which causes increased gas cost as the code can be rectified to limit the storage read to only once.

Recommendation

We advise to rectify the code on the aforementioned lines by introducing a local variable to limit the storage read of `depositLimit` to only once to save gas cost associated with the extra storage read operation.

FSP-01 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	Minor	governance/FeeSharingProxy.sol: 97	Pending

Description

The function parameters of `_protocol` and `_staking` are not validated against zero address.

Recommendation

We advise to validate the aforementioned address type parameters against zero address.

FSP-02 | Unneeded Packing of Local Variable

Category	Severity	Location	Status
Gas Optimization	Informational	governance/FeeSharingProxy.sol: 264	Pending

Description

The aforementioned lines declare local variable of type `uint32`. The data type is not packed unless it resides in the contract's storage and as EVM operates on 32-byte at a time, the uint32 has to be expanded to 32-byte (or uint256) for operations which costs more gas compared to if the aforementioned local variable is declared as type `uint256`.

Recommendation

We recommend to change the data type of `uint32` local variables on the aforementioned lines to `uint256` to save gas cost.

FSP-03 | Potential Volatile Implementation

Category	Severity	Location	Status
Logical Issue	Discussion	governance/FeeSharingProxy.sol: 334	Pending

Description

The assignment on the aforementioned line will result in discrepancy of the `numTokens` amount if in the future the withdrawal interval is removed that allows writing of checkpoints within the same block.

Recommendation

We advise to substitute the assignment with addition assignment to allow the code handle the case if withdraw interval is removed.

FSP-04 | Requisite Value of ERC-20 `transferFrom()` / `transfer()` Call

Category	Severity	Location	Status
Logical Issue	Minor	governance/FeeSharingProxy.sol: 145, 205	Pending

Description

While the ERC-20 implementation does necessitate that the `transferFrom()` / `transfer()` function returns a `bool` variable yielding `true`, many token implementations do not return anything i.e. Tether (USDT) leading to unexpected halts in code execution.

Recommendation

We advise that the `SafeERC20.sol` library is utilized by OpenZeppelin to ensure that the `transferFrom()` / `transfer()` function is safely invoked in all circumstances.

FSP-05 | Inefficient Storage Read

Category	Severity	Location	Status
Gas Optimization	Informational	governance/FeeSharingProxy.sol: 255	Pending

Description

The code on aforementioned lines read `numTokenCheckpoints[_loanPoolToken]` from contract's storage thrice which causes increased gas cost as the code can be rectified to limit the storage read to only once.

Recommendation

We advise to rectify the code on the aforementioned lines by introducing a local variable to limit the storage read of `numTokenCheckpoints[_loanPoolToken]` to only once to save gas cost associated with the extra storage read operation.

GAS-01 | Division Before Multiplication

Category	Severity	Location	Status
Mathematical Operations	Minor	governance/GovernorAlpha.sol: 529~534	Pending

Description

Mathematical operations in the aforementioned function perform divisions before multiplications. Performing multiplication before division can sometimes avoid loss of precision.

Recommendation

We recommend applying multiplications before divisions if integer overflow would not happen in functions.

GAS-02 | Potential Logic Flaw in `state()`

Category	Severity	Location	Status
Logical Issue	Medium	governance/GovernorAlpha.sol: 535	Pending

Description

The conditional `proposal.forVotes <= totalVotesMajorityPercentage` on the aforementioned line categorizes the proposal as defeated even when `forVotes` are equal to majority percentage of votes.

Recommendation

We advise to rectify the conditional such that proposal is considered defeated only when the `forVotes` are less than majority percentage of votes by using less-than operator rather than less-than or equal-to operator.

GAS-03 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	governance/GovernorAlpha.sol: 200	Pending

Description

The following functions are declared as `public`, contain array function arguments, and are not invoked in any of the contracts contained within the project's scope. The functions that are never called internally within the contract should have external visibility.

Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

GAS-04 | Comparison with `boolean` Literal

Category	Severity	Location	Status
Gas Optimization	Informational	governance/GovernorAlpha.sol: 466	Pending

Description

The aforementioned line performs comparison with literal `false` which can be substituted with the negation of the expression itself to increase the legibility of the codebase.

Recommendation

We advise to substitute the comparison with literal `false` with the negation of the expression.

GAS-05 | Inefficient Storage Read

Category	Severity	Location	Status
Gas Optimization	Informational	governance/GovernorAlpha.sol: 347, 321, 347	Pending

Description

The aforementioned line reads storage array `proposal.targets`'s length upon each iteration of the `for` loop that consumes additional gas.

Recommendation

We advise to store the `proposal.targets`'s length in a local variable before utilizing it in the conditional part of `for` loop to save gas cost associated with multiple storage reads.

GVS-01 | Usage of `transfer()` for Sending Ether

Category	Severity	Location	Status
Volatile Code	Minor	governance/GovernorVault.sol: 46	Pending

Description

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs.

Recommendation

We advise that the linked `.transfer()` and `.send()` calls are substituted with the utilization of the `sendValue()` function from the `Address.sol` implementation of OpenZeppelin either by directly importing the library or copying the linked code.

LCB-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	modules/LoanClosingsBase.sol: 48	Pending

Description

In the contract `Affiliates`, the role `owner` has the ability to set the target address for external functions of the aforementioned contract.

Any compromise to the `owner` account allows the hacker to set the target address of its choice and insert malicious code.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

LCB-02 | Dangerous Usage of `tx.origin`

Category	Severity	Location	Status
Volatile Code	Minor	modules/LoanClosingsBase.sol: 124	Pending

Description

`tx.origin` check will be no longer valid after EIP-3074 is added in the coming months. EIP3074 introduces two EVM instructions AUTH and AUTHCALL. The first sets a context variable authorized based on an ECDSA signature. The second sends a call as the authorized. This essentially delegates control of the EOA to [a] smart contract. This means there will be a way for smart contracts to send transactions in the context of an Externally Owned Account, thus bypassing this check.

Recommendation

We would recommend removal of this check as it is redundant and look into enhancing the code security for the prevention griefing attacks.

LCW-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	modules/LoanClosingsWith.sol: 47	Pending

Description

In the contract `LoanClosingsWith`, the role `owner` has the ability to set the target address for external functions of the aforementioned contract.

Any compromise to the `owner` account allows the hacker to set the target address of its choice and insert malicious code.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

LMS-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	farm/LiquidityMining.sol: 77	Pending

Description

The function `setLockedSOV` can be called by owner or a whitelisted admin to changed `LockedSOV` contract's address. As `LockedSOV` contract receives users' rewards and deals with vesting and withdrawal of user's unlocked rewards, a comprise of `owner` or any `admin` account introduces the risk of compromising user's rewards by setting a malicious `lockedSOV` address.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

LMS-02 | Redundant Statements

Category	Severity	Location	Status
Volatile Code	Informational	farm/LiquidityMining.sol: 579	Pending

Description

The function call `_updateRewardDebt(pool, user);` on the aforementioned line redundantly sets `rewardDebt` for the exiting user to zero as the `rewardDebt` is already set to zero on L575.

Recommendation

We advise to remove the redundant function call on the aforementioned line.

LMS-03 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	farm/LiquidityMining.sol: 96	Pending

Description

In the contract `LiquidityMining`, the role `wrapper` has the ability to withdraw users' funds to itself on L483.

Any compromise to the `wrapper` account may allow the hacker to take advantage of this and withdraw users' LPs to itself.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

LOS-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	modules/LoanOpenings.sol: 39	Pending

Description

In the contract `LoanOpenings`, the role `owner` has the ability to set the target address for external functions of the aforementioned contract.

Any compromise to the `owner` account allows the hacker to set the target address of its choice and insert malicious code.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

LSC-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	modules/LoanSettings.sol: 39	Pending

Description

In the contract `LoanSettings`, the role `owner` has the ability to set the target address for external functions of the aforementioned contract.

Any compromise to the `owner` account allows the hacker to set the target address of its choice and insert malicious code.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

LSD-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	modules/LoanMaintenance.sol: 71	Pending

Description

In the contract `LoanMaintenance`, the role `owner` has the ability to set the target address for external functions of the aforementioned contract.

Any compromise to the `owner` account allows the hacker to set the target address of its choice and insert malicious code.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

LSO-01 | `AdminAdded` Event Is Not Emitted

Category	Severity	Location	Status
Volatile Code	Informational	locked/LockedSOV.sol: 136~138	Pending

Description

The function `constructor` on the aforementioned line does not emit event `AdminAdded` for the admins that are added.

Recommendation

We advise to emit the event `AdminAdded`.

LSO-02 | Storage Variables Are Not Packed

Category	Severity	Location	Status
Gas Optimization	Informational	locked/LockedSOV.sol: 23	Pending

Description

The `bool` type variable on the aforementioned line occupies a complete storage slot of the contract. As the `bool` type is represented by `uint8` under the hood, we can tight pack the bool and contract type variable on L31 in a single slot by placing them next to each other.

Recommendation

We advise to place bool and contract type variables on the aforementioned lines next to each other to save gas cost associated with additional storage slot.

```
bool public migration;
IERC20 public SOV;
```

LSO-03 | Return Variable Utilization

Category	Severity	Location	Status
Gas Optimization	Informational	locked/LockedSOV.sol: 386, 395, 404	Pending

Description

The linked function declarations contain explicitly named `return` variables that are not utilized within the function's code block.

Recommendation

We advise that the linked variables are either utilized or omitted from the declaration.

LSO-04 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	locked/LockedSOV.sol: 148, 161, 176, 319	Pending

Description

In the contract `VestingRegistry`, the role `admin` has the authority over the following function:

- `addAdmin()`
- `removeAdmin()`
- `changeRegistryCliffAndDuration()`
- `startMigration()`

Any compromise to the `admin` account may allow the hacker to take advantage of this and change the sensitive parameters.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

LTA-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	connectors/loantoken/LoanTokenSettingsLowerAdmin.sol: 74	Pending

Description

The following functions are declared as `public`, contain array function arguments, and are not invoked in any of the contracts contained within the project's scope. The functions that are never called internally within the contract should have external visibility.

Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

LTA-02 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	connectors/loantoken/LoanTokenSettingsLowerAdmin.sol: 149	Pending

Description

The function parameter of `_maxScaleRate` is not validated to be less than `WEI_PERCENT_PRECISION` before being set to a state variable. This can result in wrong calculation of borrow interest rates if the `_maxScaleRate` exceeds `WEI_PERCENT_PRECISION`.

Recommendation

We advise to validate the `_maxScaleRate` parameter to be less than `WEI_PERCENT_PRECISION`.

LTA-03 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	connectors/loantoken/LoanTokenSettingsLowerAdmin.sol: 142, 74, 104	Pending

Description

In the contract `LoanTokenSettingsLowerAdmin`, the role `admin` has the authority to set up/disable parameters for specific loans and change sensitive state variables that determine interest rates on borrowing of assets. The centralization aspect of these functions are that the admin can change these parameters at will.

- `setupLoanParams()`
- `disableLoanParams()`
- `setDemandCurve()`

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

LTB-01 | Inefficient Storage Layout

Category	Severity	Location	Status
Gas Optimization	Informational	connectors/loantoken/LoanTokenBase.sol: 65~66	Pending

Description

The storage variable declared on the aforementioned line can be tight packed with variable `loanTokenAddress` on L50 by placing before it. This will result in both of the variables occupying only one storage slot resulting in reduced gas cost.

Recommendation

We advise that the variable on the aforementioned be moved before variable `loanTokenAddress` so they can be tight packed.

LTD-01 | Documentation Discrepancy

Category	Severity	Location	Status
Inconsistency	Informational	connectors/loantoken/LoanTokenLogicStandard.sol: 97	Pending

Description

The comment on the aforementioned line describes the parameter `receiver` as receiver of minted tokens yet it receives the underlying tokens upon redemption of `iTokens`.

Recommendation

We advise to rectify the comment on the aforementioned line.

LTD-02 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	connectors/loantoken/LoanTokenLogicStandard.sol: 102	Pending

Description

The function parameter `receiver` on the aforementioned line is not validated against zero address value. As the address in `receiver` parameter receives underlying loan tokens, it should be validated against zero address value.

Recommendation

We recommend to validate the `receiver` parameter against zero address value.

LTD-03 | Inefficient Storage Read

Category	Severity	Location	Status
Gas Optimization	Informational	connectors/loantoken/LoanTokenLogicStandard.sol: 218	Pending

Description

The code on aforementioned lines read `transactionLimit[collateralTokenAddress]` from contract's storage twice which causes increased gas cost as the code can be rectified to limit the storage read to only once.

Recommendation

We advise to rectify the code on the aforementioned lines by introducing a local variable to limit the storage read of `transactionLimit[collateralTokenAddress]` to only once to save gas cost associated with the extra storage read operation.

LTD-04 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	connectors/loantoken/LoanTokenLogicStandard.sol: 343~344	Pending

Description

The `require` statements on the aforementioned lines do not have error messages specified.

Recommendation

We recommend adding error messages to the `require` statements on the aforementioned lines.

LTD-05 | Return Variable Utilization

Category	Severity	Location	Status
Gas Optimization	Informational	connectors/loantoken/LoanTokenLogicStandard.sol: 1443	Pending

Description

The linked function declarations contain explicitly named `return` variables that are not utilized within the function's code block.

Recommendation

We advise that the linked variables are either utilized or omitted from the declaration.

LTD-06 | Inefficient Storage Read

Category	Severity	Location	Status
Gas Optimization	Informational	connectors/loantoken/LoanTokenLogicStandard.sol: 891	Pending

Description

The code on aforementioned lines read `liquidityMiningAddress` from contract's storage twice which causes increased gas cost as the code can be rectified to limit the storage read to only once.

Recommendation

We advise to rectify the code on the aforementioned lines by introducing a local variable to limit the storage read of `liquidityMiningAddress` to only once to save gas cost associated with the extra storage read operation.

LTD-07 | Potential Allowance Overwritten

Category	Severity	Location	Status
Logical Issue	Minor	connectors/loantoken/LoanTokenLogicStandard.sol: 1532	Pending

Description

The function call `_internalTransferFrom` on the aforementioned line passes value `minted` for allowance from `receiver` to `msg.sender`, and the token transfer amount is also the same as the value `minted`. This results in the function `_internalTransferFrom` setting allowance to zero on `L487`, effectively overriding the previously set allowance. This behavior is unexpected from the end user's perspective if there is already a non-zero allowance existing from `receiver` to `msg.sender`.

Recommendation

We advise to pass `uint256(-1)` for the allowance instead of the value `minted`, so the previously set remains unchanged.

```
_internalTransferFrom(receiver, liquidityMiningAddress, minted, uint256(-1));
```

LTD-08 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	connectors/loantoken/LoanTokenLogicStandard.sol: 1523	Pending

Description

In the contract `LoanTokenLogicStandard`, the role `owner` has the authority to change `liquidityMiningAddress`. There is centralization risk involved as users can chose to deposit their minted liquidity tokens in liquidity mining contract for farming.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

LTS-01 | Contract's State Can Be Re-initialized

Category	Severity	Location	Status
Centralization / Privilege	Major	connectors/loantoken/LoanToken.sol: 132	Pending

Description

The `initialize` function on the aforementioned line can be called by the contract `owner` multiple times. This introduces a centralization risk where contract `owner` can be change important parameters of contract such as address of loan token and the price.

Recommendation

We advise to introduce a check ensuring that `initialize` function cannot be called more than once. One possible check can be to assert that the state variable `loanTokenAddress`'s value is zero.

```
require(
    loanTokenAddress == address(0),
    "cannot be initialized twice"
);
```

LTS-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	connectors/loantoken/LoanToken.sol: 90	Pending

Description

In the contract `LoanToken`, the role `owner` has the authority to reset the `implementation` address to which the proxy contract delegates to. This introduces the centralization risk as new implementation contract can contain arbitrary code.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

MSK-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	multisig/MultiSigKeyHolders.sol: 74, 107, 169, 179, 194, 202, 257, 271	Pending

Description

The following functions are declared as `public`, contain array function arguments, and are not invoked in any of the contracts contained within the project's scope. The functions that are never called internally within the contract should have external visibility.

Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

MSK-02 | Ethereum Addresses Count Can Exceed `MAX_OWNER_COUNT`

Category	Severity	Location	Status
Logical Issue	Minor	multisig/MultiSigKeyHolders.sol: 84	Pending

Description

The function `_addEthereumAddress` on the aforementioned line adds Ethereum address for the multisig wallet. The function does not contain check to ensure that the total count of Ethereum addresses after adding the current Ethereum address does not exceed `MAX_OWNER_COUNT`, because if the total count of Ethereum addresses exceed `MAX_OWNER_COUNT` then the modifier `validRequirement` reverts on L56.

Recommendation

We recommend to introduce a check ensuring that total count of Ethereum addresses do not exceed `MAX_OWNER_COUNT`.

```
require(
    ethereumAddresses.length <= MAX_OWNER_COUNT,
    "total count of ethereum addresses exceed MAX_OWNER_COUNT"
);
```

MSK-03 | Bitcoin Addresses Count Can Exceed `MAX_OWNER_COUNT`

Category	Severity	Location	Status
Logical Issue	Minor	multisig/MultiSigKeyHolders.sol: 179	Pending

Description

The function `_addBitcoinAddress` on the aforementioned line adds Bitcoin address for the multisig wallet. The function does not contain check to ensure that the total count of Bitcoin addresses after adding the current Bitcoin address does not exceed `MAX_OWNER_COUNT`, because if the total count of Bitcoin addresses exceed `MAX_OWNER_COUNT` then the modifier `validRequirement` reverts on L56.

Recommendation

We recommend to introduce a check ensuring that total count of Bitcoin addresses do not exceed `MAX_OWNER_COUNT`.

```
require(
    bitcoinAddresses.length <= MAX_OWNER_COUNT,
    "total count of Bitcoin addresses exceed MAX_OWNER_COUNT"
);
```

MSW-01 | Inefficient Storage Struct Layout

Category	Severity	Location	Status
Gas Optimization	Informational	multisig/MultiSigWallet.sol: 39	Pending

Description

The members of the struct `Transaction` on the aforementioned line are not tightly packed in the current layout and consume one additional storage slot.

Recommendation

We advise to re-order the members of the struct by placing `bool executed;` next to `address destination;`, so both members only occupy one storage slot.

```
struct Transaction {
    address destination;
    bool executed;
    uint256 value;
    bytes data;
}
```

MSW-02 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	multisig/MultiSigWallet.sol: 9	Pending

Description

The `require` statements in the contract `MultiSigWallet` does not have error messages specified.

Recommendation

We advise to add the error messages for the `require` statements in the aforementioned contract to increase code legibility.

MSW-03 | Inefficient Storage Read

Category	Severity	Location	Status
Gas Optimization	Informational	multisig/MultiSigWallet.sol: 294	Pending

Description

The aforementioned line reads storage array `owners`'s length upon each iteration of the `for` loop that consumes additional gas.

Recommendation

We advise to store the `owners`'s length in a local variable before utilizing it in the conditional part of `for` loop to save gas cost associated with multiple storage reads.

OIC-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	governance/Vesting/OriginInvestorsClaim.sol: 109, 121, 140	Pending

Description

In the contract `authorizedBalanceWithdraw`, any authorized account, which will be evaluated by `onlyAuthorized` modifier, has the authority over the following function:

- `authorizedBalanceWithdraw()`
- `setInvestorsAmountsListInitialized()`
- `appendInvestorsAmountsList()`

Any compromise to the authorized accounts may allow the hacker to take advantage of this and potentially manipulate the project, such as withdraw `SOV` funds from the contract.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

OIC-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	governance/Vesting/OriginInvestorsClaim.sol: 90	Pending

Description

In the contract `OriginInvestorsClaim`, the role `owner` has the authority to add new admins addresses at will that introduces centralization risk as assigning of role to a new address is at the discretion of single owner address.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

PFR-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	feeds/PriceFeedRSKOracle.sol: 57	Pending

Description

In the contract `PriceFeedRSKOracle`, the role `owner` has the authority to change `rskOracleAddress`. This introduces centralization risk as this contract address is used to fetch the price on L40 and a malicious contract can return wrong result.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

PFS-01 | Redundant Variable Initialization

Category	Severity	Location	Status
Gas Optimization	Informational	feeds/PriceFeeds.sol: 48	Pending

Description

All variable types within Solidity are initialized to their default ""empty"" value, which is usually their zeroed out representation.

Particularly:

- `uint` / `int`: All `uint` and `int` variable types are initialized at `0`
- `address`: All `address` types are initialized to `address(0)`
- `byte`: All `byte` types are initialized to their `byte(0)` representation
- `bool`: All `bool` types are initialized to `false`
- `ContractType`: All contract types (i.e. for a given `contract ERC20 {}` its contract type is `ERC20`) are initialized to their zeroed out address (i.e. for a given `contract ERC20 {}` its default value is `ERC20(address(0))`)
- `struct`: All `struct` types are initialized with all their members zeroed out according to this table

Recommendation

We advise that the linked initialization statements are removed from the codebase to increase legibility.

PFS-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	feeds/PriceFeeds.sol: 319, 330, 343, 354	Pending

Description

In the contract `PriceFeeds`, the role `owner` has the authority to change the sensitive variable `protocolTokenEthPrice`. This introduces centralization risk as the owner can change protocol token's prince in Eth at will. And the role `owner` also has the authority to change the price feeds' addresses against tokens. Centralization risk involves the setting of malicious price feeds that return wrong prices.

Moreover, the role `owner` also has the authority to the following two functions. Any compromise to the `owner` account may allow the hacker to take advantage of this and manipulate the sensitive variables.

- `setDecimals()`
- `setGlobalPricingPaused()`

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

PSS-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	modules/ProtocolSettings.sol: 44, 92, 99, 108, 122, 134, 146, 159, 188, 203, 216, 229, 242, 255, 268, 281, 292, 301, 317, 488, 497, 533, 547, 561, 575, 589, 603, 645	Pending

Description

In the contract `ProtocolSettings`, the role `owner` has the ability to set the target address for external functions of the aforementioned contract.

- `initialize()`
- `setSovrynProtocolAddress()`
- `setSOVTokenAddress()`
- `setLockedSOVAddress()`
- `setMinReferralsToPayoutAffiliates()`
- `setPriceFeedContract()`
- `setSwapsImplContract()`
- `setLoanPool()`
- `setSupportedTokens()`
- `setLendingFeePercent()`
- `setTradingFeePercent()`
- `setBorrowingFeePercent()`
- `setAffiliateFeePercent()`
- `setAffiliateTradingTokenFeePercent()`
- `setLiquidationIncentivePercent()`
- `setMaxDisagreement()`
- `setSourceBuffer()`
- `setMaxSwapSize()`
- `setFeesController()`
- `withdrawProtocolToken()`
- `depositProtocolToken()`
- `setSovrynSwapContractRegistryAddress()`
- `setWrbtcToken()`
- `setProtocolTokenAddress()`
- `setRolloverBaseReward()`

- `setRebatePercent()`
- `setSpecialRebates()`
- `togglePaused()`

Any compromise to the `owner` account allows the hacker to set the target address of its choice and manipulate the entire project system.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

PSS-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	modules/ProtocolSettings.sol: 23	Pending

Description

In the contract `ProtocolSettings`, the role `owner` has the authority to change several sensitive state variables. This introduces centralization risk as the variable changeable by the owner greatly impact the Sovryn protocol. Additionally, the comment on L89-90 states that `protocolAddress` should be set only once yet there is no such check in place guarding against resetting of `protocolAddress`.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

PTU-01 | Possibility of Protocol Tokens Being Locked

Category	Severity	Location	Status
Logical Issue	Minor	mixins/ProtocolTokenUser.sol: 33	Pending

Description

The function `_withdrawProtocolToken` allow withdrawing of Protocol tokens. The aforementioned line retrieves contract's balance of Protocol tokens by reading a state variable `protocolTokenHeld` which is updated upon depositing of Protocol tokens through `depositProtocolToken` function and withdrawing of Protocol tokens through `withdrawProtocolToken` function. The use of state variable `protocolTokenHeld` to determine contract's balance does not allow withdrawing of Protocol tokens that are sent without `depositProtocolToken`, perhaps accidentally, effectively locking the in the contract.

Recommendation

We advise to utilize `balanceOf` function of Protocol token address to retrieve latest balance of the contract prior to performing transfer of withdrawing tokens. This will allow withdrawing of any directly sent Protocol tokens.

```
    function _withdrawProtocolToken(address receiver, uint256 amount) internal returns
(address, bool) {
        uint256 withdrawAmount = amount;

        uint256 tokenBalance = IERC20(protocolTokenAddress).balanceOf(address(this));
        if (withdrawAmount > tokenBalance) {
            withdrawAmount = tokenBalance;
        }
        if (withdrawAmount == 0) {
            return (protocolTokenAddress, false);
        }

        protocolTokenHeld = tokenBalance.sub(withdrawAmount);

        IERC20(protocolTokenAddress).safeTransfer(receiver, withdrawAmount);

        return (protocolTokenAddress, true);
    }
```

SED-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	modules/SwapsExternal.sol: 41	Pending

Description

In the contract `SwapsExternal`, the role `owner` has the ability to set the target address for external functions of the aforementioned contract.

Any compromise to the `owner` account allows the hacker to set the target address of its choice and insert malicious code.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

SOV-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	token/SOV.sol: 39	Pending

Description

In the contract `SOV`, the role `owner` has the authority to mint arbitrary number of `SOV` tokens to arbitrary addresses in the absence of hard cap on the token's supply.

Not only the sole owner posses centralized authority over the tokens supply, but any compromise to the `owner` account may allow the hacker to take advantage of this, and mint and dump the `SOV` tokens.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

SOV-02 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	token/SOV.sol: 55	Pending

Description

The following functions are declared as `public`, contain array function arguments, and are not invoked in any of the contracts contained within the project's scope. The functions that are never called internally within the contract should have external visibility.

Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

SRP-01 | Inheritance Order Does Not Allow Expanding of `StakingRewardsStorage` Contract With Additional Storage Structures

Category	Severity	Location	Status
Logical Issue	Medium	governance/StakingRewards/StakingRewardsProxy.sol: 13	Pending

Description

The `StakingRewards` contract inherits from `StakingRewardsStorage` and `Initializable` contracts containing state variables. The inheritance order does not allow expanding of `StakingRewardsStorage` with additional state variables if the need arises as any additional state variables introduced in `StakingRewardsStorage` will overwrite the storage of the `Initializable` contract. Additionally, the contract `StakingRewardsProxy` does not inherit from storage variables containing `Initializable` contract that can result in storage discrepancy in the future when `StakingRewardsStorage` is modified.

Recommendation

We advise to place the `Initializable` contract first and `StakingRewardsStorage` contract at the end of inheritance order for both the contracts, `StakingRewards` and `StakingRewardsProxy`, so later it can be expanded with additional state variables.

SRR-01 | Inheritance Order Does Not Allow Expanding of `StakingRewardsStorage` Contract With Additional Storage Structures

Category	Severity	Location	Status
Logical Issue	Medium	governance/StakingRewards/StakingRewards.sol: 20	Pending

Description

The `StakingRewards` contract inherits from `StakingRewardsStorage` and `Initializable` contracts containing state variables. The inheritance order does not allow expanding of `StakingRewardsStorage` with additional state variables if the need arises as any additional state variables introduced in `StakingRewardsStorage` will overwrite the storage of the `Initializable` contract. Additionally, the contract `StakingRewardsProxy` does not inherit from storage variables containing `Initializable` contract that can result in storage discrepancy in the future when `StakingRewardsStorage` is modified.

Recommendation

We advise to place the `Initializable` contract first and `StakingRewardsStorage` contract at the end of inheritance order for both the contracts, `StakingRewards` and `StakingRewardsProxy`, so later it can be expanded with additional state variables.

SRR-02 | Inefficient Storage Read

Category	Severity	Location	Status
Gas Optimization	Informational	governance/StakingRewards/StakingRewards.sol: 160~161	Pending

Description

The code on aforementioned lines read `withdrawals[staker]` from contract's storage twice which causes increased gas cost as the code can be rectified to limit the storage read to only once.

Recommendation

We advise to rectify the code on the aforementioned lines by introducing a local variable to limit the storage read of `withdrawals[staker]` to only once to save gas cost associated with the extra storage read operation.

SRR-03 | Inaccurate Validity Check

Category	Severity	Location	Status
Logical Issue	Medium	governance/StakingRewards/StakingRewards.sol: 160	Pending

Description

The conditional validation `currentTS > withdrawals[staker]` on the aforementioned line does not ensure that the claiming of rewards is happening after 14 days.

Recommendation

We advise to refactor the conditional validation and revert if the time passed since last claiming of rewards is less than 14 days.

SSD-01 | Potential Volatile Implementation

Category	Severity	Location	Status
Volatile Code	Minor	core/State.sol: 212	Pending

Description

The function `_setTarget` on the aforementioned line redundantly adds `sig` in `logicTargetsSet` when non zero targets are added successively against a particular `sig` as is the case in `initialize` implementations of `module` contracts that allow successive adding of non-zero targets for their external functions. This results in the `logicTargetsSet` containing duplicate values of `sig`.

Recommendation

We advise to refactor the `_setTarget` function such that it does not redundantly add `sig` to `logicTargetsSet` when it already exists in the set.

```
function _setTarget(bytes4 sig, address target) internal {
        address prevTarget = logicTargets[sig];
        logicTargets[sig] = target;

        if (
            prevTarget == address(0)
            && target != address(0)
        ) {
            logicTargetsSet.addBytes32(bytes32(sig));
            return;
        }

        if (
            target == address(0)
            && prevTarget != address(0)
        ) logicTargetsSet.removeBytes32(bytes32(sig));
}
```

SSS-01 | Missing Sanity Validation

Category	Severity	Location	Status
Logical Issue	Minor	governance/Staking/Staking.sol: 120	Pending

Description

The function call `_decreaseDelegateStake` on the aforementioned line decrease delegate stake for zero address when a user is staking for the first time and has no prior delegate. This adds an unnecessary checkpoint for the zero address against the specified timestamp.

Recommendation

We recommend to add a check ensuring that the variable `previousDelegatee` is not a zero address before calling `_decreaseDelegateStake`.

```
if (previousDelegatee != address(0)) {
  /// @dev Decrease stake on previous balance for previous delegatee.
  _decreaseDelegateStake(previousDelegatee, until, previousBalance);
}
```

SSS-02 | Lack of Error Message

Category	Severity	Location	Status
Coding Style	Informational	governance/Staking/Staking.sol: 182	Pending

Description

The `require` statement can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise refactoring the linked code and providing corresponding error message for better maintainability.

SSS-03 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	governance/Staking/Staking.sol: 612, 622, 663	Pending

Description

In the contract `Staking`, the `owner` has the authority over the following functions `setNewStakingContract`, `setFeeSharing`, and `unlockAllTokens`.

Any compromise to the `owner` account may allow the hacker to take advantage of the contract.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

SSS-04 | Redundant Variable Initialization

Category	Severity	Location	Status
Coding Style	Informational	governance/Staking/Staking.sol: 677, 688	Pending

Description

All variable types within Solidity are initialized to their default ""empty"" value, which is usually their zeroed out representation.

Particularly:

- `uint` / `int`: All `uint` and `int` variable types are initialized at `0`
- `address`: All `address` types are initialized to `address(0)`
- `byte`: All `byte` types are initialized to their `byte(0)` representation
- `bool`: All `bool` types are initialized to `false`
- `ContractType`: All contract types (i.e. for a given `contract ERC20 {}` its contract type is `ERC20`) are initialized to their zeroed out address (i.e. for a given `contract ERC20 {}` its default value is `ERC20(address(0))`)
- `struct`: All `struct` types are initialized with all their members zeroed out according to this table

Recommendation

We advise that the linked initialization statements are removed from the codebase to increase legibility.

SSS-05 | Inefficient Storage Read

Category	Severity	Location	Status
Gas Optimization	Informational	governance/Staking/Staking.sol: 679, 689	Pending

Description

The code on aforementioned lines read `kickoffTS` from contract's storage twice which causes increased gas cost as the code can be rectified to limit the storage read to only once.

Recommendation

We advise to rectify the code on the aforementioned lines by introducing a local variable to limit the storage read of `kickoffTS` to only once to save gas cost associated with the extra storage read operation.

STA-01 | Visibility Specifiers Missing

Category	Severity	Location	Status
Language Specific	Informational	governance/Staking/StakingStorage.sol: 26, 39, 42, 45~46, 51	Pending

Description

The linked variable declarations do not have a visibility specifier explicitly set.

Recommendation

Inconsistencies in the default visibility the Solidity compilers impose can cause issues in the functionality of the codebase. We advise that visibility specifiers for the linked variables are explicitly set.

STA-02 | Redundant Variable Initialization

Category	Severity	Location	Status
Gas Optimization	Informational	governance/Staking/StakingStorage.sol: 60	Pending

Description

All variable types within Solidity are initialized to their default "empty" value, which is usually their zeroed out representation.

Particularly:

- `uint` / `int`: All `uint` and `int` variable types are initialized at `0`
- `address`: All `address` types are initialized to `address(0)`
- `byte`: All `byte` types are initialized to their `byte(0)` representation
- `bool`: All `bool` types are initialized to `false`
- `ContractType`: All contract types (i.e. for a given `contract ERC20 {}` its contract type is `ERC20`) are initialized to their zeroed out address (i.e. for a given `contract ERC20 {}` its default value is `ERC20(address(0))`)
- `struct`: All `struct` types are initialized with all their members zeroed out according to this table

Recommendation

We advise that the linked initialization statements are removed from the codebase to increase legibility.

SVR-01 | Lack of Error Message

Category	Severity	Location	Status
Coding Style	Informational	governance/Vesting/SVR.sol: 91	Pending

Description

The `require` statement can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise refactoring the linked code and providing corresponding error message for better maintainability.

SVR-02 | Visibility Specifiers Missing

Category	Severity	Location	Status
Language Specific	Informational	governance/Vesting/SVR.sol: 27~35	Pending

Description

The linked variable declarations do not have a visibility specifier explicitly set.

Recommendation

Inconsistencies in the default visibility the Solidity compilers impose can cause issues in the functionality of the codebase. We advise that visibility specifiers for the linked variables are explicitly set.

TSD-01 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	governance/Timelock.sol: 95	Pending

Description

The function parameter of `admin_` is not validated against zero address.

Recommendation

We advise to validate the aforementioned address type parameter against zero address.

TSV-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	governance/Vesting/TokenSender.sol: 75	Pending

Description

The following functions are declared as `public`, contain array function arguments, and are not invoked in any of the contracts contained within the project's scope. The functions that are never called internally within the contract should have external visibility.

Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

TSV-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	governance/Vesting/TokenSender.sol: 56	Pending

Description

In the contract `TokenSender`, the role `owner` has the authority to add new admins addresses at will that introduces centralization risk as assigning of role to a new address is at the discretion of single owner address.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

VCV-01 | Inefficient Storage Struct Layout

Category	Severity	Location	Status
Gas Optimization	Informational	governance/Vesting/VestingCreator.sol: 25~32	Pending

Description

The struct properties `governanceControl` of type `bool` and `tokenOwner` of type `address` can be placed next to each other to tight pack them in a single storage slot.

Recommendation

We advise to observe tight packing of aforementioned storage struct properties to save gas cost associated with additional storage slot.

VCV-02 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	governance/Vesting/VestingCreator.sol: 55	Pending

Description

The function parameter `_receiver` is not validated against zero address.

Recommendation

We advise to validate the address type parameter against zero address.

VCV-03 | Comparison with `boolean` Literal

Category	Severity	Location	Status
Gas Optimization	Informational	governance/Vesting/VestingCreator.sol: 113, 126	Pending

Description

The aforementioned lines perform comparison with boolean literal which can substituted with either with the expression or the negation of the expression to save gas cost.

Recommendation

We advise to substitute the comparison with boolean literal on `L29` with the expression itself and on `L30` with the negation of the expression.

VCV-04 | Function Return Value Ignored

Category	Severity	Location	Status
Volatile Code	Informational	governance/Vesting/VestingCreator.sol: 139	Pending

Description

The return values of function call `SOV.approve(address(vesting), vestingData.amount);` on the aforementioned line is ignored.

Recommendation

We advise to handle the return value of the function call on the aforementioned line.

VCV-05 | Potential Stuck Vesting Process

Category	Severity	Location	Status
Volatile Code	Medium	governance/Vesting/VestingCreator.sol: 112, 125, 156	Pending

Description

The contract has a possibility of being stuck when a vesting is creating through `processVestingCreation` call setting `vestingCreated` state variable to `true`, if the vesting is not processed and an authorized account removes the vesting from the list then the contract's logic does not allow creating of new vesting with `processVestingCreation` always reverting the call.

Recommendation

We advise to move the statement `vestingCreated = false;` on `L142` outside of the `if` block, such that the lock can be freed by a call to `processStaking`.

VCV-06 | Inefficient Usage of Local Storage Variable

Category	Severity	Location	Status
Gas Optimization	Informational	governance/Vesting/VestingCreator.sol: 128	Pending

Description

The aforementioned line create a local storage variable of type `VestingData` struct. As the same properties of storage struct are read several times in the code following its declaration, this results in increased gas cost from repeated storage read.

Recommendation

We advise to change the data location of aforementioned local variable from `storage` to `memory` to save gas cost.

```
VestingData memory vestingData = vestingDataList[vestingDataList.length - 1];
```

VCV-07 | Inefficient Storage Read

Category	Severity	Location	Status
Gas Optimization	Informational	governance/Vesting/VestingCreator.sol: 127	Pending

Description

The code on aforementioned lines read `vestingDataList.length` from contract's storage thrice which causes increased gas cost as the code can be rectified to limit the storage read to only once.

Recommendation

We advise to rectify the code on the aforementioned lines by introducing a local variable to limit the storage read of `vestingDataList.length` to only once to save gas cost associated with the extra storage read operation.

VLV-01 | Lack of Error Message

Category	Severity	Location	Status
Coding Style	Informational	governance/Vesting/VestingLogic.sol: 80	Pending

Description

The `require` statement can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise refactoring the linked code and providing corresponding error message for better maintainability.

VLV-02 | Inefficient Storage Read

Category	Severity	Location	Status
Gas Optimization	Informational	governance/Vesting/VestingLogic.sol: 101	Pending

Description

The aforementioned lines repeatedly perform storage read `endDate` upon each iteration of `for` loop, which results in significant gas cost.

Recommendation

We advise to utilize a local variable and intialize it with `endDate` and then use local variable in the conditional of `for` loop.

VRL-01 | Inheritance Order Does Not Allow Expanding of `VestingRegistryStorage` Contract With Additional Storage Structures

Category	Severity	Location	Status
Logical Issue	Medium	governance/Vesting/VestingRegistryLogic.sol: 11	Pending

Description

The `VestingRegistryLogic` contract inherits from `VestingRegistryStorage` and `Initializable` contracts containing state variables. The inheritance order does not allow expanding of `VestingRegistryStorage` with additional state variables if the need arises as any additional state variables introduced in `VestingRegistryStorage` will overwrite the storage of the `Initializable` contract. Additionally, the contract `VestingRegistryProxy` does not inherit from storage variables containing `Initializable` contract that can result in storage discrepancy in the future when `VestingRegistryStorage` is modified.

Recommendation

We advise to place the `Initializable` contract first and `VestingRegistryStorage` contract at the end of inheritance order for both the contracts, `VestingRegistryLogic` and `VestingRegistryProxy`, so later it can be expanded with additional state variables.

VRL-02 | Redundant `modifier` Usage

Category	Severity	Location	Status
Coding Style	Informational	governance/Vesting/VestingRegistryLogic.sol: 117~119	Pending

Description

The function `createVesting` on the aforementioned line is redundantly guarded by the `onlyAuthorized` modifier as the function internally calls `createVestingAddr` which is already guarded by the same modifier.

Recommendation

We advise to remove the redundant use of modifier `onlyAuthorized` with the function `createVesting` to increase code legibility.

VRL-03 | Return Variable Utilization

Category	Severity	Location	Status
Gas Optimization	Informational	governance/Vesting/VestingRegistryLogic.sol: 291, 299	Pending

Description

The linked function declarations contain explicitly named `return` variables that are not utilized within the function's code block.

Recommendation

We advise that the linked variables are either utilized or omitted from the declaration.

VRS-01 | Usage of `transfer()` for Sending Ether

Category	Severity	Location	Status
Volatile Code	Minor	governance/Vesting/VestingRegistry2.sol: 183	Pending

Description

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs.

Recommendation

We advise that the linked `.transfer()` and `.send()` calls are substituted with the utilization of the `sendValue()` function from the `Address.sol` implementation of OpenZeppelin either by directly importing the library or copying the linked code.

VRS-02 | Redundant Variable Initialization

Category	Severity	Location	Status
Coding Style	Informational	governance/Vesting/VestingRegistry2.sol: 288	Pending

Description

All variable types within Solidity are initialized to their default ""empty"" value, which is usually their zeroed out representation.

Particularly:

- `uint` / `int`: All `uint` and `int` variable types are initialized at `0`
- `address`: All `address` types are initialized to `address(0)`
- `byte`: All `byte` types are initialized to their `byte(0)` representation
- `bool`: All `bool` types are initialized to `false`
- `ContractType`: All contract types (i.e. for a given `contract ERC20 {}` its contract type is `ERC20`) are initialized to their zeroed out address (i.e. for a given `contract ERC20 {}` its default value is `ERC20(address(0))`)
- `struct`: All `struct` types are initialized with all their members zeroed out according to this table

Recommendation

We advise that the linked initialization statements are removed from the codebase to increase legibility.

VRV-01 | Usage of `transfer()` for Sending Ether

Category	Severity	Location	Status
Volatile Code	Minor	governance/Vesting/VestingRegistry.sol: 210, 234	Pending

Description

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs.

Recommendation

We advise that the linked `.transfer()` and `.send()` calls are substituted with the utilization of the `sendValue()` function from the `Address.sol` implementation of OpenZeppelin either by directly importing the library or copying the linked code.

VRV-02 | Redundant Variable Initialization

Category	Severity	Location	Status
Coding Style	Informational	governance/Vesting/VestingRegistry.sol: 358	Pending

Description

All variable types within Solidity are initialized to their default ""empty"" value, which is usually their zeroed out representation.

Particularly:

- `uint` / `int`: All `uint` and `int` variable types are initialized at `0`
- `address`: All `address` types are initialized to `address(0)`
- `byte`: All `byte` types are initialized to their `byte(0)` representation
- `bool`: All `bool` types are initialized to `false`
- `ContractType`: All contract types (i.e. for a given `contract ERC20 {}` its contract type is `ERC20`) are initialized to their zeroed out address (i.e. for a given `contract ERC20 {}` its default value is `ERC20(address(0))`)
- `struct`: All `struct` types are initialized with all their members zeroed out according to this table

Recommendation

We advise that the linked initialization statements are removed from the codebase to increase legibility.

VRV-03 | Inefficient Storage Read

Category	Severity	Location	Status
Gas Optimization	Informational	governance/Vesting/VestingRegistry.sol: 198~199, 334~335	Pending

Description

The code on aforementioned lines read `lockedAmount[msg.sender]` from contract's storage twice which causes increased gas cost as the code can be rectified to limit the storage read to only once.

Recommendation

We advise to rectify the code on the aforementioned lines by introducing a local variable to limit the storage read of `lockedAmount[msg.sender]` to only once to save gas cost associated with the extra storage read operation.

VRV-04 | Explicitly Returning Local Variable

Category	Severity	Location	Status
Gas Optimization	Informational	governance/Vesting/VestingRegistry.sol: 219	Pending

Description

The function on the aforementioned line explicitly returns local variable which increases overall cost of gas.

Recommendation

Since named return variables can be declared in the signature of a function, consider refactoring to remove the local variable declaration and explicit return statement in order to reduce the overall cost of gas.

VRV-05 | Inefficient Storage Read

Category	Severity	Location	Status
Gas Optimization	Informational	governance/Vesting/VestingRegistry.sol: 328, 359	Pending

Description

The aforementioned lines repeatedly perform storage read `CSOVtokens.length` upon each iteration of `for` loop, which results in significant gas cost.

Recommendation

We advise to utilize a local variable to store `CSOVtokens.length` and then utilize local variable in the conditional part of `for` loop to save gas cost.

VRV-06 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	governance/Vesting/VestingRegistry.sol: 158~159, 167, 233, 249, 266, 286, 297, 313	Pending

Description

In the contract `VestingRegistry`, the role `admin` has the authority over the following function:

- `addAdmin()`
- `removeAdmin()`
- `withdrawAll()`
- `setVestingFactory()`
- `setCSOVtokens()`
- `setBlacklistFlag()`
- `setLockedAmount()`
- `transferSOV()`

Any compromise to the `admin` account may allow the hacker to take advantage of this and change the sensitive parameters, and transfer all the ETH and/or SOV tokens to any arbitrary address.

Recommendation

We advise the client to carefully manage the aforementioned account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to 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 the different level in term of short-term and long-term:

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key.
- Introduction of a DAO/governance/voting module to increase transparency and user involvement when assigning privileged roles to addresses.

WSS-01 | Inefficient Storage Read

Category	Severity	Location	Status
Gas Optimization	Informational	governance/Staking/WeightedStaking.sol: 378, 384~385	Pending

Description

The aforementioned lines repeatedly perform storage read `kickoffTS` which results in significant gas cost.

Recommendation

We advise to utilize a local variable to the address `kickoffTS` and then utilize it in the aforementioned lines to save gas cost associated with repeated storage reads.

Appendix

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.