Flare Network

Smart Contract Audit

coinspect

Flare

coinspect

Flare Network Launch

Smart Contract Audit

V220530

Prepared for Flare • May 2022

1. Executive Summary

In **May 2022**, **Flare** engaged Coinspect to perform a source code review of **Flare Network**. The objective of the project was to continue evaluating the security of the smart contracts.

The code changes and additions in scope for this audit did not introduce any security vulnerabilities to the existing system. Coinspect identified certain scenarios where the new governance implementation could be manipulated, but these threats do not exist in the context in which they are intended to be used in Flare as clarified by the team, and their risk level was downgraded accordingly to reflect this fact.

The following issues were identified during the initial assessment:

High Risk	Medium Risk	Low Risk
0	0	5
Fixed 0	Fixed 0	Fixed 4

2. Assessment and Scope

The audit started on **May 16, 2022** and was conducted on the **flare_launch_audit** branch of the git repository at https://gitlab.com/flarenetwork/flare-smart-contracts/ as of commit **34fd94142a19364a71e6b94ed2062e86f3ba65e7** of **May 17, 2022**.

The final version of the repository reviewed was commit **4002620da4c5b3735cbdb1faa5a483cb26a7ddb2** of May 26, 2022 which merged updated tests for the new features in scope.

A third version of the repository with the fixes provided by the Flare team was reviewed, with commit **f14a38675ebbb0606df2b12f3217ec09fc78db8e** as of **June 1, 2022**.

The audited files have the following sha256sum hash:

```
df615639d224c56efbffe23432fd960f0613dd3594ed24ac9da32c995d96771c  tokenPools/mock/DistributionMock.sol
a6b0e1cc627f69bf3875064d01627d2a4d7de74ef4d4ce78c3bc2575a47070e3  tokenPools/implementation/FtsoRewardManager.sol
68e536215e580bc0073a401087a09f0921f2a75c9c62e7295ca1803cf31506f1  tokenPools/implementation/UnearnedRewardBurner.sol
10c07fcdd7abf30940fcb0a3f4269b8404ec9b64a6ece4896802d90516828390  tokenPools/implementation/DistributionTreasury.sol
1d91ec36010303a1868d5a8baf69d4d73cb1ae4ebbab98ab599524ee2b9df42d  tokenPools/implementation/Distribution.sol
4d2a54ec5be6764fb1050f0e866368c8422e9f8434d60cda62b122e8e6bb5c19  tokenPools/implementation/TeamEscrow.sol
e637ee40c52c081169b4342859bbd8dfff386cdbdad8199eb9429a1679185ff2  tokenPools/implementation/DistributionToDelegators.sol
3884c7b7ed7e82baaa92572a21e1551546a15667c46b069bd0e22cfb6c9d4e4a  tokenPools/interface/IITokenPool.sol
239b1a4cea7f3097af27803b9abdd97d26bd5bf126abbcee2f2b9e6e2610888c  tokenPools/interface/IIFtsoRewardManager.sol
78e989283e1b423761a7621a889553f1fc60de9fc3647058f3dc8aec8d548937  personalDelegation/mock/CloneFactoryMock.sol
5788b4a62ab1b5561c6002bf2c14b94d1ef8f4b89daca388003ac8a537f3c9dd  personalDelegation/implementation/DelegationAccountManager.sol
220330021c85cc21af937a388fba46eae3587d5677fa684fb86cc30b359e714  personalDelegation/implementation/CloneFactory.sol
8ca8ee3f67b79ce531cec5ec02e657b088c7ab1a57e843b06531df529cc69356  personalDelegation/implementation/DelegationAccountClonable.sol
d8f9b8e089fa05f14140e9bba7fd1d37b533b514f5444342e5e07c844f70b0bc  personalDelegation/interface/IDelegationAccountManager.sol
32b1b27211b2f04d2fd0bfc648f7ed014c7ab1c458654df8bdaa3b31db860d8b  personalDelegation/interface/IDelegationAccount.sol
ea8a2ae0d3a53555929103d1006bd9e33bc361b5c6193e3b5749448383c4ad53  mockXAsset/mock/AssetToken.sol
a74a32cc14ad2134c626485068e5c76b3071bc5057ced810a966757745bfe329  mockXAsset/mock/DummyAssetMinter.sol
e048e4b7718b3070b7bce54987b84330318703286ddd0c79d7ab738515eac52f  mockXAsset/interface/ICollateralizable.sol
aea5d38cb92435e56b8601b5f3fd7bee5062da993feebd249c71b9bb6481f1a8  genesis/mock/InfiniteLoopMock.sol
6b40d53e3dfbc29c43aa039797d59f61773b4ace0053c3f800157bab8ce808a0  genesis/mock/FlareDaemonMock.sol
e65c5fb83e4fbfb7110111960f4f2d00fdb51f075301dbaab7b77faf8593791e  genesis/mock/TestableFlareDaemon.sol
debdeba7884368523c7022e3e73731cfc5aa5e53f24c76163b692a3eb5a5ae99  genesis/mock/FiniteLoopMock.sol
711f9844fc81676a95d498c1d6497cd125fa6627ba98dcd52ce987add9577900  genesis/mock/FlareDaemonMock1.sol
6a0183517977afc120b627432f8dfc6d8b5eb2be17d312ef791c2d3b803908e9  genesis/mock/ReadGasLeft.sol
2060606587d0518f3397e3bc0d7aa582b1c0eb4bbdf7b64cc2e339bab61d2ccb  genesis/mock/EndlessLoopMock.sol
3f0cc4a73ff5cba07c18186a9cb44558e348f5a43d243facd8a96b706e1b34d0  genesis/mock/FlareDaemonMock2.sol
3f6eff7ec1ab34756f26e822a7c17eb24578a35ac2c2bafee013b9e438e20db0  genesis/mock/InfiniteLoopMock1.sol
c8a77ece8c7eb5eb117520cd84a8beec52ff7b1866db75e8d69e76b641fef730  genesis/implementation/StateConnector.sol
f1086624d70a9569b936b547e4dcf0a7bb5318c68bafced29d8a0a631b34e5f3  genesis/implementation/PriceSubmitter.sol
e6947b14b2e464a893b5370e29123fce651e7c89ecb8fe0205ba033df12e1c9f  genesis/implementation/FlareDaemon.sol
435152b27625dbc1564bb275e819e11d65f540d183361a04061406eaecf19f00  genesis/interface/IFtsoRegistryGenesis.sol
90cac97dd9882b6f77e4a1a90dd9a812c10d8266b41c6287a8d4285d6cad0314  genesis/interface/IInflationGenesis.sol
4352cd4e2d3644b5ba19b261a6b0c65d804dc0f4fe2c64910c5aa88ffc358430  genesis/interface/IIPriceSubmitter.sol
cca145c8c6770699574f7db27662f1d76eb33d6b54a0315fdac396fae4747c76  genesis/interface/IFlareDaemonize.sol
06a7b4f3c2b67af0a92094f6e53ced08da3f8abadd3859f1e1bfcf8e70fe8c05  genesis/interface/IFtsoGenesis.sol
907c5c8b95ff8dccdd8572c63654716fe9f1d626df43520f1a5995bbd388019f  genesis/interface/IFtsoManagerGenesis.sol
888dee1a4d362fcb2c0f154f56116dec520f0b23e126942f52813a32aa790811  addressUpdater/mock/AddressUpdatableMock.sol
fb74132581784ebcd120221c5478b26ddb915bb0a5707856cc52bf50412d1770  addressUpdater/implementation/AddressUpdater.sol
6cf01dd4544e028cd562bb0ccd9dc1f1c350b9ae2fc6dc3d96b3afd28f20fd95  addressUpdater/implementation/AddressUpdatable.sol
59824abf5524c24e54c161f1faeed77763eab397b4eb7d02304291fbc63d4abc  addressUpdater/interface/IIAddressUpdatable.sol
b8a3b3fd8b284b92f1d19492c57b7c091e5869b8c2b18a2cade797e0f2bf7d83  governance/mock/ExecuteMock.sol
```

```
f1b03df0e5b8863aed1cb97c5b0ec7bef51837713c14f18b4a64e3974f01ccbe  governance/mock/HistoryCleanerMock.sol
50cf903c3c490096555d32cc0f2d7db032e555e4ccad461ac1b95e88699cf7da  governance/implementation/GovernorAcceptSettings.sol
7881cab5d26c78094ec9368cb266cfe27e5ea8f4005b288056636b48c6683ae8  governance/implementation/GovernorVotePower.sol
972ff7d926b7a7109d19b9ef5f4e1311e1a8c7f5e1cc7c0b20dd9d76686f677b  governance/implementation/GovernorReject.sol
46fc4851512562ea274b0823547a3a2deea7d70cc8ec992ce7f0d1eac74eec7c  governance/implementation/GovernorRejectSettings.sol
6fbba549f7aed180002e0ac541cc747ed7e7e0b59ad1c304718e869e874135de  governance/implementation/GovernorAccept.sol
af67366f14783742254274d98259e85eb45998ec103cc002818568493e6c5969  governance/implementation/GovernorProposals.sol
fb582ea6980b1adb4d78ebf6a83e5a522e68fa990f961feda28291dfca6fea5a  governance/implementation/GovernedBase.sol
400aa7be810a08c979234d5756cd6dbd5c85ed4948cdc6220e772146feed303b  governance/implementation/Governor.sol
93cf605f3233de4520b4e27474677166106690aa8e2563e135d5d98720356700  governance/implementation/Governed.sol
e0d147fb7158a7a866ccae28bca80296f032dc8db1e9fc40e8262a725a8bc7eb  governance/implementation/GovernedAtGenesis.sol
f486c41ea986e089937dddbe5e77a4b9d0271d22edeabc314f092aaeaab2f62f  governance/implementation/GovernorSettings.sol
2fbe896ace28ed9b860be8fe708b804e43ea9907a11fd8cbd4b8120a8c66e443  governance/implementation/GovernorVotes.sol
10f1d191c4748bcdc5498d1c8370f7b482a40505ed56e8723d350830df9e27ae  governance/interface/IIGovernor.sol
45a72e36d5d1e146b563a3ce000c1cc9082287b59c64053522e1baa006c018d5  governance/interface/IIGovernorAccept.sol
b4d30716f7389b58d29b4e96dfdf1360f2ccdc1fd845524aaf1c2a85c13c157e  governance/interface/IIGovernorReject.sol
8fce6d8cb170c9dc4927af657fc008ab123355995acb0c8a939897b2d678a124  userInterfaces/IFtso.sol
22b2f694b7bf7eb2fd447785b74343346a77003edb9db6fdaa2ea79932debff4  userInterfaces/IVPContractEvents.sol
8d4a1b9a377e116d1704d53541d6c86b2a7608e2fad78d898b1a6b8817d166e5  userInterfaces/IGovernor.sol
fd3111ed7cbaa101d4bb85eb79e05469c6ad20355f66149b15a2935002b44cb7  userInterfaces/IPriceSubmitter.sol
bddea7532299880f0a76e9067266d4fa18477826096326c7c1bf448e84c6a0b4  userInterfaces/IWNat.sol
b850523a4bf8173ca747030b4db8a9118ef2992a7264f6afbbf509623c0aa13c  userInterfaces/IVPToken.sol
238f3ff859014b80a973879d4a7382d5a2a5c38db8d9a5895fe7dae76560a898  userInterfaces/IDistribution.sol
5b358284a55bf4024d3bfba16d4af593f6558c13162e0885234c640dbdb85366  userInterfaces/IValidatorRegistry.sol
482aaa07addf41346e3996b31edd592c7cba28b95e0f0ebd0ebeefae79b8f8ed  userInterfaces/IFtsoRegistry.sol
8df8c58009f85cea7822e077eb94b0576858534eb2e13204a4c342405ac2563f  userInterfaces/IGovernanceVotePower.sol
9f80875e92cb3067aabb251ad746136b3e3f55c5fa9f129643132fb487196b35  userInterfaces/IFtsoRewardManager.sol
43d8624273b06f2ef77a277b5ad98b1c3fe7ae89edf6c627711a26e4b2b94074  userInterfaces/IFtsoManager.sol
eaa082812a3486b88dd4cbc25c1ead80fc0d3993dfa7c4c2669c0842c49245d9  userInterfaces/IVoterWhitelister.sol
356bbdceba273eabdcdb44836c77b60af9bc8a316fdcb2a87b167822ad6e4819  userInterfaces/IDistributionToDelegators.sol
27235ab0ac08281c05f39130bf12ad8237cd5850fee2fd64bfbfa200b673e876  inflation/mock/InflationReceiverMock.sol
c251acd5490b7e08e5fdedd0ae2ab4fab0905645a0455f2407af85e55c00896d  inflation/mock/InflationMock.sol
5e17fe844f3db062e3bfa397254ba17b6cc6e56f5d6b3462073941b671a9ae4a  inflation/mock/PercentageProviderMock.sol
483d4d4e050ebcf4dfbcc04a9e20a0a7fdaf287d920c98ea128713d4727af9af  inflation/mock/SuicidalMock.sol
04b20a9bf8e179038d98d15711f1cb37c9292e5aff538f8d2022c40af8511a52  inflation/mock/InflationMock1.sol
77e4fd4021fae07fa17b34b026293f2ce76d6e076c665bbf25d7dc22b3c646f0  inflation/mock/FlareDaemonWithInflationMock.sol
234730acd6c7a35fb1d69672c164c6d6360e738544dfdd8f21e7b254266a91c3  inflation/mock/InflationReceiverAndTokenPoolMock.sol
1bc6273bbf6c5b530ab803f6dab3a08b56ad90e228e3234763eec8acab8f7220  inflation/implementation/Inflation.sol
8a771c8a894e5441d7190e51d6f8a17475c3d3f0da288c7e8a586ad516110321  inflation/implementation/Supply.sol
40dda7efa90cd1db1d8f813205b718b9dbd8d0e9057375dffd37a1ef7db580f4  inflation/implementation/InflationAllocation.sol
6cfa3d870184d95635066a4a243e7ee16784d8d1ae74b991a4046af8de617950  inflation/lib/InflationAnnum.sol
d0df108753a5d4aa4351266a547f2db3ea27019ac9a18034d9b58f443c167cf9  inflation/lib/InflationAnnums.sol
942dd3f1315811deb28deb14dfdaf47225e396733bc99b3391f4e240f5d7e3d2  inflation/lib/RewardServices.sol
b87a9d8f63b2ef7cdf0d507339d6f2a6497378a84940269f8156d24597adec71  inflation/lib/RewardService.sol
a29709dcdb968b1d91cfba5e354eb8f54314668e834de300dcd69a5187bf9b15  inflation/interface/IIInflationAllocation.sol
2382188fe84843eec1dc34043c145fed21c9afc5f584473790c546c80b4de46e  inflation/interface/IISupply.sol
093ce091266460e8f4d4682c3d235ae63ae20590dd3f1fdf052ee03c752586ac  inflation/interface/IIInflationReceiver.sol
1f19fc317f93cadaaf8937db4fcb61aa17f4949694e7cec9a7ab958278c31733  inflation/interface/IIInflationPercentageProvider.sol
32a518978197b162f36bbcee446200d6424e126c8077b93ee9c7ace449a7ad0d  inflation/interface/IIInflationSharingPercentageProvider.sol
b13a5fe247a4b80022cbeb5a63f371de391f3a6da8947476075d7a7c95abf8d9  utils/mock/PriceSubmitterUnregisterHack.sol
8987b038958572bbd87aae6862ac2ec3770aa158037a1ff4523767b3bf1691af  utils/mock/SafePctMock.sol
ef603d230385fbefec6d97312ac1f2e9d73beed653654e04cbb942b1f3b6fc1c  utils/mock/VoterWhitelisterMock.sol
6945f0b2926288db2a94f7c91fd30fec36f57337423853b47c5f825099cda3dc  utils/mock/GasConsumer3.sol
6da309494640bd043d894407fce4c7dae1948fc289a8b66afac5e2587dd027cc  utils/mock/GasConsumer.sol
e2e45ea0659ab4b848ba5fb9f79c99d519db95da524901232cdd96670ea3c616  utils/mock/GasConsumer5.sol
ec1a3852a028d30cc2e8ff18206110b46fcc1ee1f345a7e512e28f0696c94baa  utils/mock/GasConsumer7.sol
83da6848fc51feec6b3165f7f2917d74eb3dc19b7ea32262b13d7c2322edc5ef  utils/mock/GasConsumer2.sol
f90c7c91a6c17c2fa1a5821671fd21e5fb303a79721709cfcd70ff88591d849b  utils/mock/GasConsumer6.sol
0ea9b26b1502d7507ed8a9ddea625718d24b3830749d80c71a077393541e01fc  utils/mock/DateTimeLibraryContractMock.sol
7456379c2cc2e3ba2856c7798ed6ab6e4f802030cda14e4dbf4ac55b04d4b30d  utils/mock/GasConsumer4.sol
0aa255d92f4f87c1d4ad4066574e08bfd9350dd0352f6188d9680ee0c072458b  utils/implementation/PriceReader.sol
c3fd3d454ce3d09c90c22c58631febdb2e5b17499e2c269775f393d5675a59e7  utils/implementation/VoterWhitelister.sol
2c134d5f194559ad9e4087297bbd669a22c20c7f76aaf93bde833e262682795c  utils/implementation/RevertErrorTracking.sol
da7d13381925ca65b5b184d651fb0a5af86a7146b4bf5b8735c64a51bc51bb68  utils/implementation/DateTimeLibrary.sol
0dacb948c6a6011d0493f9e8992cadf09080bb6d5773636dfdad85562cf542bf  utils/implementation/SafePct.sol
bce891791f2cf868f98158ee333e41c53df1ebb138c661dac836472386c56512  utils/implementation/FtsoRegistry.sol
a81c7e3cbda4eb930d004f48745a7030c35a694fd78579b416c31d8f6f7d43ef  utils/implementation/GovernedAndFlareDaemonized.sol
3617f640c89459bb1f020fbc0f6f4adc0e2c574735970af0586b651618661ac4  utils/Imports.sol
37c6eafd11bbcb296df58c3234410ca756617aa61506b188d43c73db8230d4b0  utils/interface/IIVoterWhitelister.sol
c45be59283140525575dabf7d5a108625d29736ef42374612d279aa73aca0ef1  utils/interface/IUpdateValidators.sol
d7971865bc3504135040b25015c8204e41ca8d3ed4209e096dd21e9f7458b04f  utils/interface/IIFtsoRegistry.sol
b1e1ba0244b65518678392368908705608f8c265f558b301a4b3db86c044a208  ftso/mock/FtsoMedianMock.sol
```

```
d41de91655a84a7b11eb63d1a263dde2f27ac27786a1a149a2709c827799c6ac  ftso/mock/MockUpdateValidators.sol
73f800dbda665dac03e9ff4f0da07846ac420e4606babab39b0094dbc12c43af  ftso/mock/MockFtso.sol
1245adac9a228015d4a6dadd6c999cd26e978874593eb08b85163da4abba88ea  ftso/mock/FtsoManagerMock.sol
5f813d421bc347975d69feb95a924a49d828e95839b8d0110061e7a96a42834e  ftso/mock/MockVPToken.sol
9183a2ed51eceba778da2d281b3189fc1815cc9202807f2b9c5d94f4f192fecb  ftso/mock/FtsoEpochMock.sol
85cddafd066db0ba9aa6bc79672c918d0e64907c5573f6b3b5fc0437aec81264  ftso/mock/FtsoManagerV1Mock.sol
f66861232f386413afcf177147de2023d559e7fa433b6795591730c602f2510c  ftso/mock/SimpleMockFtso.sol
e6c05d055ccbebcba1038124f1d1e29b19fe737d1f5842e5a52f8a36a6433be7  ftso/mock/FtsoVoteMock.sol
2cf6b476f3d7571e6a2d2c6e8124e83e1ca34c3da62e369c48f1340fdf05ed9b  ftso/implementation/Ftso.sol
446d6b454e0d9a8cb5847ca07442038482e9319d549143aaf84fadeda2ab64da  ftso/implementation/FtsoManager.sol
7bf51497be17b9f262780999b35175db42b16cbedc1bda573ea85f55661c0d5c  ftso/priceProviderMockContracts/PriceProviderMockContracts.sol
345bd77c8440f060dc7476afda383ec3853122ba88b273efcd2ef396447af680  ftso/priceProviderMockContracts/priceProviderMockFtso.sol
6147dd29e7b0ad02bb1351020d9fbcc0d79b6091f948b43f8a7c4c132a03ba51  ftso/lib/FtsoVote.sol
7bd96a3f5e0efb9bbbe173c8008d25909070a171365917cd3b731d4d046b2a21  ftso/lib/FtsoMedian.sol
65d75474bd8bdc4734375c70879a024bacad09c44abe337bf67cfaf1b47d6504  ftso/lib/FtsoEpoch.sol
610f6e35ef8d0f7635546da682e0b85066f6d7da531e42b521550a8823143feb  ftso/lib/FtsoManagerSettings.sol
e7b809c8cf5c193506aede6477b61b1eafb7f62311c5773e59385a74351294f0  ftso/interface/IIFtsoManager.sol
2dbd881f3182805649dacbf119ffc054d148f315144dbe12938b1074894eff08  ftso/interface/IIFtso.sol
c5cdbdf3da71c72202233f06c0cfcd484adb064add6b467a8e281812ba13ad9a  ftso/interface/IIFtsoManagerV1.sol
795ed7e1637f23eacf1a80da16aa2eb4785e65d2f9cac0c4d89a6b3f651fa9bf  token/mock/VPTokenMock.sol
f9a26c45723f7222eb7fb92c03559b23f20393559b0a3a4be2bcb552f189757a  token/mock/DummyVPToken.sol
dce2c78c18163d8540271e4a4b74cafe680e80f8ac853442a81d847b7b5099c5  token/mock/TransferToWnatMock.sol
4c94b4f56769094663535fd042148691e5736ceaea8e87fcd59bd2f1cb5b919c  token/mock/FlashLoanMock.sol
841498726cdf0115b452fc8f35a657a6ee2d21a1e775a88ee7ea7f733b9b33ff  token/mock/CheckPointsByAddressMock.sol
cea31e015368633710f9857b298c7f1a6ecfee141c2a30c4cd5cdf2d8e98e532  token/mock/PercentageDelegationMock.sol
666fe341bf0b0a5f0e28d7d339581eb89adb117fc095f99bd66d7dc37b3a5940  token/mock/CheckPointHistoryMock.sol
9f9022acb627b70ba075e57bb1aa978bbcb3b4d07ceb9be7e2acf075294a4e41  token/mock/VotePowerMock.sol
8a06b88a399e75a55a404fb8fbcc970db8643f6b47a5ce4ab24a306dc1eb4d3c  token/mock/ExplicitDelegationMock.sol
4246d9082408e18033c1d2d5d6776242878a4cdeca24c696dd7e787a8f7fa8ec  token/mock/VPContractMock.sol
85476a8b8d03782f81331edbd5527513f70b2d887222f68339d66fd7729d9057  token/mock/DelegatableMock.sol
91bdd6388216144771835c99cff1c6974d35d36ee5d1fb5a017066ee835b6c86  token/mock/CheckPointableMock.sol
cdf463a2c26f7245f3678565cb1275a5ffefd633fa329e19c1e84e6af333d853  token/implementation/CheckPointable.sol
04a3bc74d57bbf9ab84cabfa308b8381db225db7a210a579f01319e6a3f89fff  token/implementation/VPToken.sol
5003b4781ad9ad55718dbb6607ec6691078a8994779c15cf507fec52565e4a7e  token/implementation/Delegatable.sol
7b188c487fb0619ea037cf6d6da8e55181d0338795f36242af108ec278592810  token/implementation/WNat.sol
6b18c3c39097c659979825df30f912e80f4e9bd518e0b8b149d0ed8b4836c1dc  token/implementation/CleanupBlockNumberManager.sol
23f612bdabff273dbff98deadcea84fe9fba7e3f2ab31c6804484f2a6904200c  token/implementation/VPContract.sol
c9cb3734108b2d34318685db5418926efef852e0da20768d927de39f87d1c634  token/implementation/GovernanceVotePower.sol
53aa232b7f50bb17d650419207735c18c7e90bddedbef074bc80e240c1442d01  token/lib/DelegationHistory.sol
796d9b62af94c15d89b94676d4514396e8a8c165c9e870e099996b3a0488cd63  token/lib/CheckPointHistoryCache.sol
511f3fd7c6694faae72aa10870b790eb72ee9e66d9a72da2b87b63fc7e788459  token/lib/DelegateCheckPointHistory.sol
5ac8ce671c7964516af30d0f8fa9633f744ef0d7e1dbd85f20840ff5f60e53eb  token/lib/DelegateCheckPointsByAddress.sol
7c7858e64134e57a649b54e5e04153f448d346e246b84923d4339c4fb576751f  token/lib/ExplicitDelegation.sol
82da0011307842f905d1b3809457ad58a0e869b7c0590a2edab9f6011e1f3c35  token/lib/PercentageDelegation.sol
ed07ecbf063aafb852b4e67baf68d7a02485144db31ed37765cdab9b9e2396bf  token/lib/VotePowerCache.sol
8ec06551b24efb1a3aef75ec0af76eca1f2845bad71eeafadd71ed514c0ab116  token/lib/VotePower.sol
8b964e7efa5b4ddb469251fb8b455f65964ed187f82707ff7eadc089ed41f8e6  token/lib/CheckPointsByAddress.sol
10f17a1c451bc43f55ba0a8a81a9a213dc1df6ec2bb7200cc21d13f422ed34dd  token/lib/CheckPointHistory.sol
63856382a0837b1e219b5f5e49a402a90877d1e4b9d3469ade5add2613fd4b8a  token/interface/IICleanable.sol
36b63687636354521b2131f9264d8ac49df62fc010d1396fb946f148c7013c27  token/interface/IIGovernanceVotePower.sol
c3f1fdbd8e3b1c298552586f4c07fc1bdd28c45d1d30dd765c3c0443afab3ca6  token/interface/IIVPContract.sol
37d9c059841c5c00853b6407cbb82cd15a8d5e144e7a0ecc039c7f818c0ccd94  token/interface/IIVPToken.sol
```

The contracts are compiled using Solidity version 0.7.6. The contracts are written clearly and tested. The project has 1171 passing tests with a line and branching coverage of above 95%.

This report is focused on the changes introduced since the last audit performed by Coinspect. These changes are briefly described below.

New airdrop distribution. The new airdrop distribution will benefit the holders of the wrapped FLR token. The distribution will start with a flat 15% total distribution and the remaining 85% will be distributed during the following 36 months (30 days periods). Every month each user will be entitled an amount proportional to the held value. This will be done by selecting 3 checkpoints during the previous month and distributing based on the values at those times. **Blocks during the first week cannot be eligible for checkpoints**.

New Treasury. This contract is used by the distribution. It is responsible for holding the tokens for the airdrop process.

New governor contracts. There are two new governor contracts (one for the canary chain and the other for the mainchain). These contracts called `GovernorAccept` and `GovernorReject` will be used as a polling mechanism for governance decisions.

Escrow contract. Some selected users (e.g. team members) should opt out from the airdrop contract and use this contract instead. It will distribute tokens on a fixed-amount basis.

Inflation update. Inflation now is based on the current supply and updated every 30 days instead of yearly.

New delegations accounts. Users can delegate the responsibility of claiming the rewards from the `FTSO` and airdrop to an account. This makes some processes simpler for the user, allowing them to save the values on a cold wallet while using a hot wallet for claiming all the rewards on a single transaction, among other benefits.

State connector. There were small changes on the attestation logic.

The new Governor contracts are only intended for voters to show intent and the automatic execution of the approved proposals will not be employed. **The final execution of proposals will always be manually performed by the Governance multisignature.** Coinspect recommends removing the execute function from the Governor contracts if it is not planned to be used and changing the contract names to reflect the intended polling purpose or **limiting the execute method to the Flare's multisignature wallet**. The last alternative would limit the Flare's ability to execute privileged proposals to those approved by the community.

The governor contracts are susceptible to a variety of attacks: selecting power blocks, bribing voters, forcing users to spend gas for rejecting contracts. Because of the context in which the governor contracts are going to be utilized in the Flare Network, these attacks were not fully evaluated during this audit, as they can not be exploited. Flare team acknowledged the possibility of these issues being abused if the intended context changes, and discussed potential mitigating factors and implementation improvements that could be added in the future. Coinspect strongly suggests the automated governance is fully evaluated in the context of Flare Network if this feature wants to be enabled.

As with other contracts, the deployment setting values have a significant impact on the security of the contracts.

3. Summary of Findings

Id	Title	Total Risk	Fixed
FLR-11	Unbounded loop in critical code path (optOutAddresses)	Low	✔
FLR-12	Governance voting power calculated from past holdings enables profiting via collusion/bribing schemes	Low	✔
FLR-13	GovernanceReject can keep making proposals to force users to waste gas in order to reject them	Low	✔
FLR-14	Attackers can brute-force the vote power block selection randomness	Low	✓
FLR-15	Missing check to prevent users sending funds to contract	Low	✔
FLR-16	Proposals ID could be duplicated on different networks	Info	✔
FLR-17	setClaimBalance could be forbidden after entitlement start time	Info	✔
FLR-18	Unused _execute should be removed from Governor	Info	✔
FLR-19	Misleading governor contract names	Info	✔
FLR-20	Governor quorum is ineffective	Info	✔

4. Detailed Findings

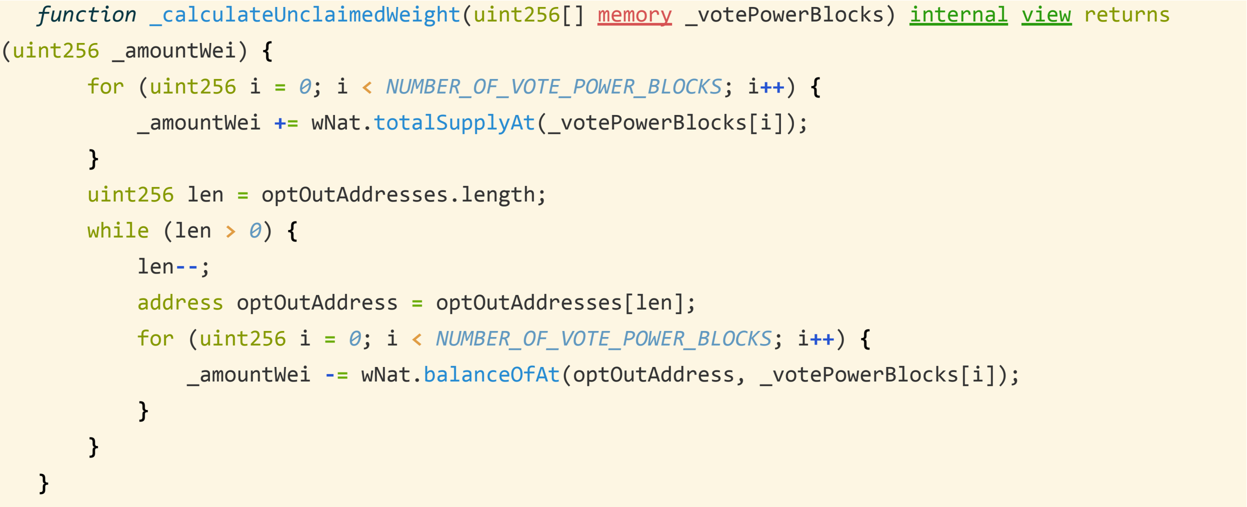
FLR-11 Unbounded loop in critical code path (optOutAddresses)

Total Risk	Impact	Location
Low	Medium	`DistributionToDelegators.sol`
Fixed ✔	Likelihood Low	

Description

The `_calculateUnclaimedWeight` function iterates over a dynamic array of opted out addresses. The `getTokenPoolSupplyData` depends on this function, and is called by the critical daemonized `Supply` contract. If the array is allowed to grow, this would force the critical contract to revert because of an out of gas condition.

The calculations performed for each address employ binary search (`_indexOfGreatestBlockLessThan` in the `CheckpointHistory` contract).



```
  function _calculateUnclaimedWeight(uint256[] memory _votePowerBlocks) internal view returns
(uint256 _amountWei) {
        for (uint256 i = 0; i < NUMBER_OF_VOTE_POWER_BLOCKS; i++) {
            _amountWei += wNat.totalSupplyAt(_votePowerBlocks[i]);
        }
        uint256 len = optOutAddresses.length;
        while (len > 0) {
            len--;
            address optOutAddress = optOutAddresses[len];
            for (uint256 i = 0; i < NUMBER_OF_VOTE_POWER_BLOCKS; i++) {
                _amountWei -= wNat.balanceOfAt(optOutAddress, _votePowerBlocks[i]);
            }
        }
    }
```

Because the `optOutAddresses` array can only grow when an opt out is accepted and confirmed by the Governance multisig, this issue would be hard to exploit. Unless, for example, an automated process is utilized to confirm opt out requests.

Recommendation

Document this potential issue so the `optOutAddresses` array length is never allowed to grow over the limit.

Status

The Flare team is fully aware of this potential problem and already has off-chain code able to calculate gas expense for this function before accepting a new opt-out address. This potential issue is what led to their implementation of the user opt-out request plus `Governance` accept flow.

FLR-12	Governance voting power calculated from past holdings enables profiting via collusion/bribing schemes	

Total Risk
Low
	Impact
	Low
		Location
		`GovernorVotePower.sol`
		`GovernanceVotePower.sol`

Fixed ✓	Likelihood Low	

Description

The user's voting power is calculated from a random block in the past. Because users' funds are not locked when the execution of their proposals occur, it is possible for them to vote for a proposal against the protocol's best interests.

Users could transfer their funds out of the Flare network before their harming proposals are executed. It would be possible for attackers to profit from this by colluding with other users (e.g., bribing scheme) to concentrate enough voting power to propose and accept a malicious governance proposal.

The necessary voting power to perform this attack will depend on the accept threshold configured in the contract. **It is important to consider that the voting power is calculated in relation to the total wrapped native token supply. Users are able to wrap and unwrap their holdings with no limit. This permits the manipulation of the wrapped supply.**

It is worth noting that the `GovernanceAccept` contract has no whitelist to propose or execute.

This issue was not fully researched because the contract automated execution feature is not intended to be used by Flare.

Recommendation

Beware of these attack scenarios if the intended purpose of these contracts or their context changes in the future. Clearly document the risks for any third party interested in relying on the voting results.

Status

This issue was acknowledged by the Flare team. However, the issue's impact was downgraded to low because the execution of the accepted proposals will not be automatic, but will be performed manually by the Governance multisignature. In this context, the ability to manipulate voting does not result in an automatic profit for the attackers.

Flare team further clarified: *These attacks would (maybe) be possible, if wrapped supply is very low compared to available non-wrapped coin supply. E.g. if we have total supply 100m, and wrapped supply 20m, then an attacker could temporarily borrow 20m+ FLR and wrap them for a short time and then return, then it would get a lot of relative vote power to go over the threshold, which is currently based on the wrapped token supply (being 100% of votes). So it is important to deploy and start using the accept contract only when enough tokens are wrapped and the 50% loan attack is practically impossible. In case this condition is fulfilled then the contracts are secure modulo 50%-of-supply-short-term-loan attack.*

To prevent the above mentioned "unwrap attack", we have two supply parameters on the network:
TCS - total circulating supply (FLR)
TWS - total wrapped supply
*We could impose condition that the proposal is valid only if in the vote power block TWS >= k * TCS, where k describes the required share of TWS in regard to TCS that ensures satisfactory size of TWS. On Songbird we have TCS = 10bln, TWS = 6bln, which gives us current k = 0.6.*
So setting k is basically the decision when we have "enough" of TWS.
Since we have multisig governance we could proceed from here in one of two ways

1. *Declare the proposal to be valid only if TWS >= k * TCS for the proposals vote power block. If this is not the case, multisig will not be executed. This is just a rule, no code needed.*
2. *Build in the parameter k into the GovernanceAccept contract and prevent the creation of the proposal if TWS < k * TCS.*

FLR-13	GovernanceReject can keep making proposals to force users to waste gas in order to reject them

Total Risk Low	Impact Low	Location `GovernanceReject.sol`
Fixed ✔	Likelihood Low	

Description

Proposals are considered accepted unless the community rejects them. As a result, a proposer can keep making new proposals, forcing users to either waste gas rejecting the proposals, or letting them pass by not voting.

This is dependent on the deployment settings, in particular the quorum threshold.

Recommendation

Coinspect recommends documenting this potential threat in case the `GovernanceReject` contract is used in a different context in the future.

Status

This issue impact and likelihood were downgraded as the (trusted) Foundation will be the only whitelisted address able to propose.

Flare team explained the `GovernanceReject` contract is intended to be used in the Songbird network (canary testnet) only.

FLR-14	Attackers can brute-force the vote power block selection randomness	
Total Risk Low	Impact Low	Location `Governor.sol`
Fixed ✓	Likelihood Low	

Description

An attacker with no investment in the platform can obtain big voting power.

Users can submit the (almost) same governance proposal multiple times until they get the desired voting power block. As a consequence they will be able to concentrate more voting power to help them pass their proposal.

Voting power blocks are evenly distributed, so users must get voting power for some estimated blocks using short loans.

The attack would work as follows:
- User takes a big loan in block N and holds WFLR for P blocks, then returns it.
- The same user (it could be with another account), submits multiple times the same proposal until the any block between N and N+P is selected
- User has more vote power than it should for the proposal

Recommendation

Because the execution of the proposals will be limited to a multisig held by the Flare team, there is no specific recommendation for this issue. Further analysis is required if the Flare team considers allowing the automatic execution of approved proposals.

Status

The issue was acknowledged by the Flare team. However, this issue's impact was downgraded to low because the **execution of the accepted proposals will not be automatic**, but will be performed manually by the Governance multisignature.

June 7, 2022: Polling contracts now possess a `wrappingThreshold`, which is a minimum amount of wrapped Flare needed to make a proposal. Flare team considers that using a high enough value this risk is mitigated, as the team stated:

> *...credibility of the attack now depends on choice of parameters, especially wrappingThreshold. If set high enough, this is very hard to achieve.*

This issue status was updated to partially fixed as the last modification is enough to prevent exploitation in certain scenarios, but the root cause has not been fully addressed. Further analysis is required to fully rule out all scenarios, including some of the components of the platform that have not been audited or deployed yet.

FLR-15 Missing check to prevent users sending funds to contract

Total Risk	Impact	Location
Low	Low	`DistributionToDelegators.sol` `Distribution.sol`
Fixed ✓	Likelihood Low	

Description

In the `Distribution` and `DistributionToDelegators` contracts the receive functions do not restrict the funds origin to the `DistributionTreasury`. This could result in users mistakenly sending funds to the contract.

```
/**
  * @notice Needed in order to receive funds from DistributionTreasury
  */
receive() external payable {
    /* empty block */
}
```

Recommendation

Add a check to only allow the `Treasury` contract to fund this contract.

Status

June 7, 2022: Only the treasury can send funds to the contracts.

FLR-16	Proposals ID could be duplicated on different networks	
Total Risk **Info**	Impact -	Location `Governor.sol`
Fixed ✓	Likelihood -	

Description

`Governor` proposals do not include a chain ID. As a consequence two proposals could result in the same proposal ID in different networks, and this could confuse off-chain code unaware of this fact.

Also, the `Governor` contract address is not included, and this could be confusing if multiple `Governor` contracts co-exist in the same network.

This is a known issue documented in the OpenZeppelin contract:
`https://github.com/OpenZeppelin/openzeppelin-contracts/blob/c4f76cfa155ec28a66bdd9bf24890174fef5b976/contracts/governance/Governor.sol#L128`

Recommendation

Consider adding chain ID and the `Governor` contract address to the proposal ID.

Status

June 7, 2022: The chain id is part of the proposal id

FLR-17	setClaimBalance could be forbidden after entitlement start time	
Total Risk **Info**	Impact -	Location `Distribution.sol`
Fixed ✔	Likelihood -	

Description

The `Distribution.setClaimBalance` function allows Governance to reset accounts airdrop balance. The current implementation enables Governance to modify an address airdrop balance after the original distribution.

Recommendation

Consider disabling the `setClaimBalance` function after the entitlement start time.

Status

June 7, 2022: `setClaimBalance` reverts once the `entitlementStartTs` value is set.

FLR-18 Unused _execute should be removed from Governor

Total Risk	Impact	Location
Info	-	`Governor.sol`
Fixed ✔	Likelihood -	

Description

The `Governor` contract includes the ability to execute accepted proposals. However, in Flare network the proposals are only intended to show voters intent and will be executed by the Governance multisig.

It is recommended to remove the inherited unused functionality from the contracts.

Recommendation

Remove the proposal execution functionality if it is not intended to be used.

Status

June 7, 2022: the method is left in the contract so users can use it for their own purpose.

FLR-19 Misleading governor contract names

Total Risk	Impact	Location
Info	-	`Governor*`
Fixed ✓	Likelihood -	

Description

The Governor contracts will not be utilized to execute proposals, but this will be the responsibility of the Governor multisig instead. The current contracts names are confusing, as the Governor contracts have no actual ability to act as governors.

Recommendation

Consider renaming the contracts to reflect the fact that they are only intended as a non-binding polling mechanism.

Status

June 7, 2022: contract names start with Polling instead of Governor. Still some contracts and functions names start with Governor, but are unlikely to be misleading.

FLR-20 Governor quorum is ineffective

Total Risk Info	Impact -	Location `GovernorAccept.sol`
Fixed ✔	Likelihood -	

Description

The `quorumThresholdBIPS` is ineffective for defining whether a proposal's votes should be counted. There are two cases to consider:

1. `quorumThresholdBIPS < acceptanceThreshold:` In this case the only important property is whether the `voteFor` value suprasses or not the `acceptanceThreshold.` If the quorum is not reached, then the `acceptanceThreshold` is also not reached, so the proposal is rejected. If the `acceptanceThreshold` is reached, then it is also the `quorumThreshold`. In either case it is only meaningful whether the `acceptanceThreshold` is reached.
2. `quorumThresholdBIPS > acceptanceThreshold:` In this case the only effect of voting against the proposal is to increase the likelihood of reaching the quorum, so none will vote against it. Generally any negative vote has no positive effect in the `GovernorAccept` contract (i.e. none will ever reasonably cast a negative vote).

One of the consequences of this design is that the `acceptanceThreshold` should be greater or equal to 50%. This is an imposed restriction based on the current design. An alternative design is that only positive votes count for reaching quorum and once the quorum is reached, the majority between negative and positive votes wins.

Recommendation

There are two alternative designs:

1. Remove the quorum and have an acceptance threshold only
2. Remove the acceptance threshold and if the quorum is reached, let the majority between `voteFor` and `voteAgainst` win.

Status

June 7, 2022: the quorum and acceptance logic have changed. Now the quorum for the `PollingAcceptance` requires reaching an absolute threshold, and then it needs to surpass a relative threshold against the against votes. A similar scenario applies to the `PollingReject` contract.

5. Disclaimer

The information presented in this document is provided "as is" and without warranty. The present security audit does not cover any off-chain systems or frontends that communicate with the contracts, nor the general operational security of the organization that developed the code.