



COINSPPECT
You build, we defend.



Smart Contract Audit
Pre-register & FTSO
Jan, 2025



Voter pre-register and FTSO Management Group

Smart Contract Audit

Version: v250109

Prepared for: Flare

January 2025

Security Assessment

1. Executive Summary
2. Summary of Findings
 - 2.3 Solved issues & recommendations
3. Scope
4. Assessment
 - 4.1 Security assumptions
 - 4.2 Decentralization
 - 4.3 Testing
 - 4.4 Code quality
5. Detailed Findings




FLRM-01 - Quorum is diluted when members are chilled after being added to the management group

6. Disclaimer

1. Executive Summary

In **December, 2024**, **Flare** engaged **Coinspect** to perform a Smart Contract Audit of the new Pre-Registry feature and FTSO Management Group. The objective of the project was to evaluate the security of the smart contracts.

The mentioned features enable pre-registration during a reward epoch for the next reward epoch for voters that are registered in the current reward epoch. The FTSO Management Group is a revamped version of the Polling FTSO smart contract adapted to support FTSO V2 system.

 Solved	 Caution Advised	 Resolution Pending
High 0	High 0	High 0
Medium 1	Medium 0	Medium 0
Low 0	Low 0	Low 0
No Risk 0	No Risk 0	No Risk 0
Total 1	Total 0	Total 0

During the security assessment, Coinspect identified the following issues: FLRM-01 shows how Management Groups of the new polling system are not updated if a voter is chilled after being added, which allows misbehaving accounts to keep participating of the voting process.

2. Summary of Findings

This section provides a concise overview of all the findings in the report grouped by remediation status and sorted by estimated total risk.

2.3 Solved issues & recommendations

These issues have been fully fixed or represent recommendations that could improve the long-term security posture of the project.

Id	Title	Risk
FLRM-01	Quorum is diluted when members are chilled after being added to the management group	Medium

3. Scope

The scope was set to be the repositories:

- Flare Smart Contracts at <https://gitlab.com/flarenetwork/flare-smart-contracts-v2> from 05d8d92052b3ebb1104df1e4773f3a3928cdd00b to 5c88d3847f0bd4117844bcbe2272d9bb610e32bc.
- Flare System Client at <https://gitlab.com/flarenetwork/flare-system-client> from 90498b9e4e329a90cb4d730a14cd0a0f0f1e7bc6 to ae69f635e6a3b1fd05cd200106a80b51253d9c63.

4. Assessment

Flare added a new smart contract that aims to improve the voter registering process. This new smart contract, named `VoterPreRegistry` allows voters that are active in the current signing policy to pre-register for the next epoch.

The main goal of this smart contract is to increase the system's stability by allowing voters to pre-register before the new epoch starts in case there is downtime when the new epoch starts. To process and allow a pre-registration, the smart contract requires voters to be registered on the current signing policy, and validates their signature. Once these steps are validated, the voter is added to a pre-registering list.

When the upcoming epoch starts, the Flare Systems Manager smart contract tries to add all pre-registered voters to the new epoch's voters list inside the `daemonize()` call. This call occurs at the beginning of each block. The process of adding each voter is executed via `try-catch` logic, to prevent abuse and halting the `daemonize` call in case a voter cannot be added to the new voters list.

To support the pre-registration process, the Flare Team modified the System Client allowing voters to submit the pre-registration signature before a new epoch starts as an opt-in feature. On top of adding this new functionality, the System Client logging was improved.

Moreover, the `PollingFTSO` smart contract was removed and replaced by the `PollingManagementGroup` contract. This adapted version of the older polling smart contract includes changes to support the FTSO V2 system.

The FTSO Management Group smart contract implements a self-regulatory framework within the FTSO ecosystem to monitor and address potential misconduct among data providers. Through a committee of qualified providers who maintain active participation and consistent performance, the system establishes a structured process for addressing infractions, requiring public discussion in the Flare FTSO Self-Policing Forum followed by formal on-chain voting that demands both a 66% quorum and majority support.

The system employs a graduated penalty structure where first-time infractions result in temporary suspension for two FTSO reward epochs, while second violations trigger permanent exclusion from the whitelist, handled by an off-chain component out of this audit's scope. Under the oversight of the Flare Foundation, this governance structure operates independently on both Flare and Songbird networks, specifically targeting behaviors that compromise ecosystem integrity such as coordinated price submissions and duplicate operations across nodes.

Membership in the management group requires providers to demonstrate sustained performance by earning rewards over 20 consecutive epochs and maintaining a clean disciplinary record without recent chilling penalties. Members can face removal if they fail to earn rewards for two consecutive epochs or demonstrate insufficient voting participation, specifically missing votes in half of the most recent four proposals that achieved quorum. Additionally, the Flare Foundation retains the authority to directly add or remove members as needed, ensuring active and responsible participation in the governance system.

4.1 Security assumptions

For this assessment, Coinspect made the following assumptions:

- Each voter decides using the System Client parameters whether to pre-register or not for the following epoch
- The Flare Foundation acts honestly respecting what is stated on the FIP-2

4.2 Decentralization

The pre-registering process heavily relies on the flare daemon to fulfill the registration process for the upcoming epoch. Moreover, the FTSO Management and voting process requires the Flare Foundation to execute the outcome of a proposal. The [FIP-2](#) claims that the Flare Foundation reserves the right to not act upon the results of the voting.

4.3 Testing

Test cases were added to evaluate the behavior of the newly added components. Coinspect was able to quickly write new scenarios thanks to the updated testing suite.

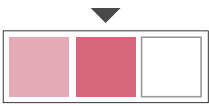
4.4 Code quality

Coinspect observed that the code quality is high, including relevant comments and NatSpec when appropriate. This documentation allows users, developers and any other stakeholder to understand what a smart contract, function and variable does.

5. Detailed Findings

FLRM-01

Quorum is diluted when members are chilled after being added to the management group

Status Solved	Risk Medium
	
Resolution Fixed	Impact Medium Likelihood Medium

Location

flare-smart-contracts-
v2/contracts/governance/implementation/PollingManagementGroup.sol

Description

Voters can be chilled after they are added to the member list and keep operating at the management contract. As a consequence, they are still allowed to submit proposals and participate on the voting of ongoing proposals.

The PollingManagementGroup smart contract restricts adding accounts to the member list if they were chilled for the last 20 blocks:

```
// check if voter was chilled in last reward epochs
if (voterRegistry.chilledUntilRewardEpochId(bytes20(voter)) +
    addAfterNotChilledEpochs >= currentRewardEpoch) {
    revert("recently chilled");
}
```

After an account is added to the management group, the system considers they can propose and cast votes:

```
function _addMember(
    address _voterToAdd,
    uint256 _currentRewardEpoch
)
    internal
{
    managementGroupMembers.add(_voterToAdd);
    // id of the last created proposal
    memberAddedAtProposal[_voterToAdd] = idCounter;
    memberAddedAtRewardEpoch[_voterToAdd] = _currentRewardEpoch;
    delete memberRemovedAtTs[_voterToAdd];
    emit ManagementGroupMemberAdded(_voterToAdd);
}
```

```
function _canPropose(address _voter) internal view returns (bool) {
    return managementGroupMembers.index[_voter] != 0;
}
```

```
function _canVote(address _voter, uint256 _proposalId) internal
view returns (bool) {
    Proposal storage proposal = proposals[_proposalId];
    return proposal.isEligible[_voter];
}
```

Where the eligibility is determined upon proposal creation if the account is part of the management group:

```
for (uint256 i = 0; i < members.length ; i++) {
    proposal.isEligible[members[i]] = true;
}
```

As a consequence, if an account is chilled *after* they are added to the management group, they can still perform all the before mentioned actions in spite of the fact they were punished. Moreover, the specification does not consider this scenario as a condition to remove a member.

Coinspect considers the likelihood of this issue to be medium since it is possible that an active member of the management group is chilled after being added. The impact is also medium since it requires multiple chilled accounts already added to the group to effectively impact on an ongoing voting process and the group maintainer can still manually remove the conflictive voters.

Recommendation

Allow anyone to remove a voter from the management group if they were chilled for the last 20 blocks.

Status

Fixed on commit [09bedaa50569718d80f18977067b9bb361de5722](#).

A new condition was added to `removeMember` that allows to remove a member if they were recently chilled.

Proof of Concept

The following test shows how the chilling status of an account does not affect their privileges to participate on an ongoing voting process at the Polling smart contract.

```
function testCoinspect_ChilledProviderCanStillVote() public {
    testPropose();

    // voters did not vote yet
    for (uint256 i = 0; i < 4; i++) {
        assertEq(pollingManagementGroup.hasVoted(1, voters[i]), false);
    }
    (uint256 votesFor, uint256 votesAgainst) =
    pollingManagementGroup.getProposalVotes(1);
    assertEq(votesFor, 0);
    assertEq(votesAgainst, 0);
    assertEq(uint256(pollingManagementGroup.state(1)), 1);

    // move to voting period -> proposal is active
    vm.warp(123 + 3600);
    assertEq(uint256(pollingManagementGroup.state(1)), 2);

    // At some point, voter 1 is chilled for 1000 epochs
    _mockChilledUntilRewardEpochId(voters[0], 1000);

    // voters 1 and 2 vote (in favor)
```

```

vm.prank(voters[0]);
vm.expectEmit();
emit VoteCast(voters[0], 1, 1, 1, 0);
pollingManagementGroup.castVote(1, 1);
vm.prank(voters[1]);
vm.expectEmit();
emit VoteCast(voters[1], 1, 1, 2, 0);
pollingManagementGroup.castVote(1, 1);
for (uint256 i = 0; i < 2; i++) {
    assertEq(pollingManagementGroup.hasVoted(1, voters[i]), true);
}
(votesFor, votesAgainst) =
pollingManagementGroup.getProposalVotes(1);
assertEq(votesFor, 2);
assertEq(votesAgainst, 0);

// voter 3 votes against; threshold is reached and majority is in
favor
vm.prank(voters[2]);
vm.expectEmit();
emit VoteCast(voters[2], 1, 0, 2, 1);
pollingManagementGroup.castVote(1, 0);
(votesFor, votesAgainst) =
pollingManagementGroup.getProposalVotes(1);
assertEq(votesFor, 2);
assertEq(votesAgainst, 1);

// move to the end of the voting period
vm.warp(123 + 3600 + 7200);
assertEq(uint256(pollingManagementGroup.state(1)), 4);
}

```

6. Disclaimer

The contents of this report are provided "as is" without warranty of any kind. Coinspect is not responsible for any consequences of using the information contained herein.

This report represents a point-in-time and time-boxed evaluation conducted within a specific timeframe and scope agreed upon with the client. The assessment's findings and recommendations are based on the information, source code, and systems access provided by the client during the review period.

The assessment's findings should not be considered an exhaustive list of all potential security issues. This report does not cover out-of-scope components that may interact with the analyzed system, nor does it assess the operational security of the organization that developed and deployed the system.

This report does not imply ongoing security monitoring or guaranteeing the current security status of the assessed system. Due to the dynamic nature of information security threats, new vulnerabilities may emerge after the assessment period.

This report should not be considered an endorsement or disapproval of any project or team. It does not provide investment advice and should not be used to make investment decisions.