FYEO

Secure Code Review of Solidity Smart Contracts on the Flare Network

Flare Networks Ltd.

November 2022
Version 1.0

Presented by:

FYEO Inc.
PO Box 147044
Lakewood CO 80214
United States

Security Level
Public

Table of Contents

List of Figures

LIST OF TABLES

EXECUTIVE SUMMARY

OVERVIEW

Flare Networks Ltd. engaged FYEO Inc. to perform a Secure Code Review of Solidity Smart Contracts on the Flare Network.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on October 10 - October 19, 2022, and focused on the following objectives:

- To provide the customer with an assessment of their overall security posture and any risks that were discovered within the environment during the engagement.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
- To identify potential issues and include improvement recommendations based on the results of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the FYEO Security Team took to identify and validate each issue, as well as any applicable recommendations for remediation.

KEY FINDINGS

The following issues were identified during the testing period. All issues have since been remediated:

- FYEO-FLARE-01 – DelegationAccountManager - Result of `createClone` is not checked
- FYEO-FLARE-02 – DelegationAccountManager - Updated WNat not reflected in DelegationAccountCloneable
- FYEO-FLARE-03 – DelegationAccountClone - Missing event in `transferExternalToken`
- FYEO-FLARE-05 – DelegationAccountManager - Missing events
- FYEO-FLARE-08 – FtsoRewardManager - Missing events
- FYEO-FLARE-09 – FtsoRewardManager - Optimization in `_isRewardClaimable`
- FYEO-FLARE-10 – FtsoRewardManager - Unclear usage of `totalSelfDestructReceivedWei`

Based on our review process, we conclude that the reviewed code implements the documented functionality.

SCOPE AND RULES OF ENGAGEMENT

The FYEO Review Team performed a Secure Code Review of Solidity Smart Contracts on the Flare Network. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

The source code was supplied through a private repository at **https://gitlab.com/btblock-cybersec/multichain/flare/flare-smart-contracts** with the commit hash e31c3a54576b48d5e5b6c80be6a1473b722dd80d.

The re-review focused on changes in the commit hash 3fab6d99d0227cd6d14356f3cdf1923e0fa2b95c

Files included in the code review
```
flare-smart-contracts/
└── contracts/
    ├── personalDelegation/
    │   └── implementation/
    │       ├── CloneFactory.sol
    │       ├── DelegationAccountClonable.sol
    │       └── DelegationAccountManager.sol
    └── tokenPools/
        └── implementation/
            ├── AttestationProviderRewardManager.sol
            ├── Distribution.sol
            ├── DistributionToDelegators.sol
            ├── Escrow.sol
            ├── FtsoRewardManager.sol
            ├── GenericRewardManager.sol
            ├── IncentivePool.sol
            ├── IncentivePoolAllocation.sol
            └── ValidatorRewardManager.sol
```

Table 1: Scope

TECHNICAL ANALYSES AND FINDINGS

During the Secure Code Review of Solidity Smart Contracts on the Flare Network, we discovered:

- 1 finding with MEDIUM severity rating.
- 1 finding with LOW severity rating.
- 5 findings with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

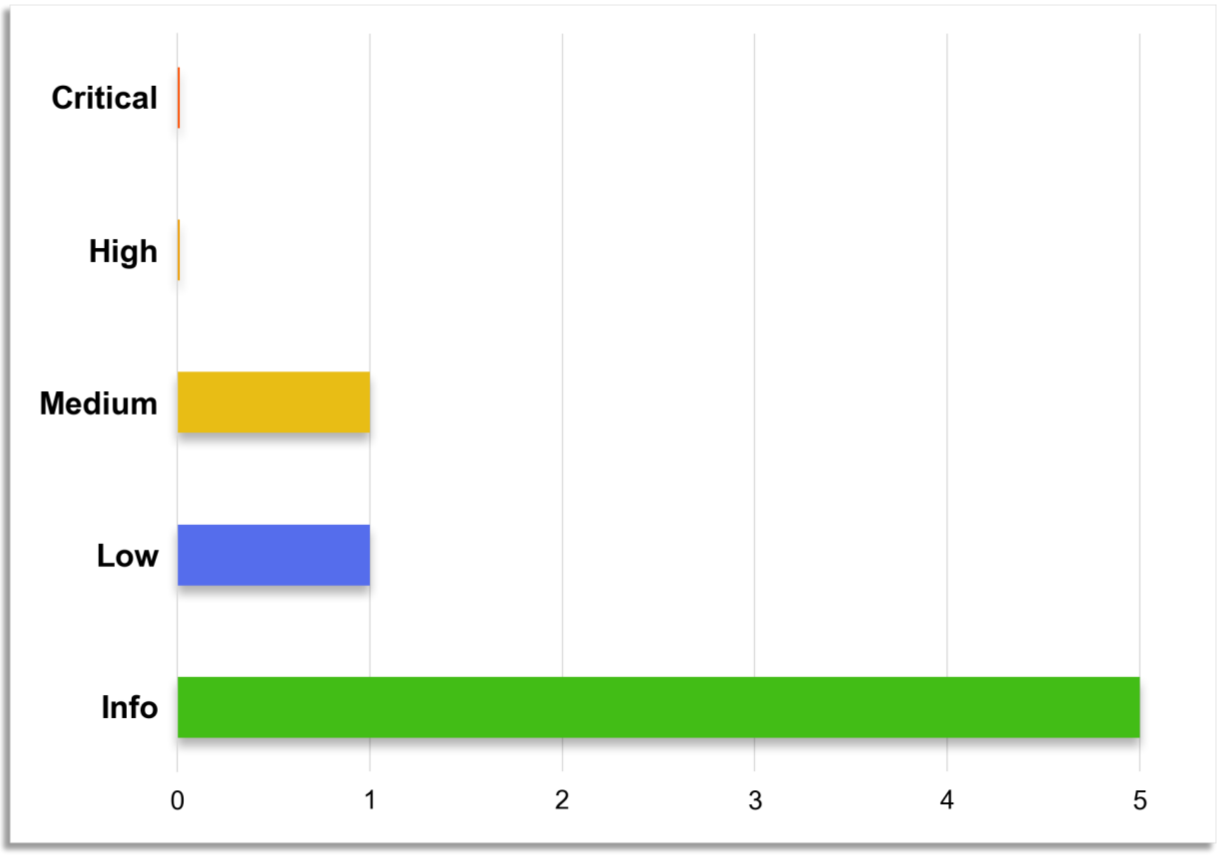


Figure 1: Findings by Severity

FINDINGS

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

Finding #	Severity	Description
FYEO-FLARE-01	**Medium**	DelegationAccountManager - Result of `createClone` is not checked
FYEO-FLARE-02	**Low**	DelegationAccountManager - Updated WNat not reflected in DelegationAccountCloneable
FYEO-FLARE-03	**Informational**	DelegationAccountClone - Missing event in `transferExternalToken`
FYEO-FLARE-05	**Informational**	DelegationAccountManager - Missing events
FYEO-FLARE-08	**Informational**	FtsoRewardManager - Missing events
FYEO-FLARE-09	**Informational**	FtsoRewardManager - Optimization in `_isRewardClaimable`
FYEO-FLARE-10	**Informational**	FtsoRewardManager - Unclear usage of `totalSelfDestructReceivedWei`

Table 2: Findings Overview

TECHNICAL ANALYSIS

The source code has been manually validated to the extent that the state of the repository allowed. The validation includes confirming that the code correctly implements the intended functionality.

TECHNICAL FINDINGS

GENERAL OBSERVATIONS

The contracts audited, provide a clever solution to reward distributions, thoroughly aggregating rewards by providers and epochs. The level of security of the reward distribution also depends on the usage of the contracts, i.e. the integrity and validity of inputs passed.

We have tried to identify the assumptions made, presented in the findings below. After discussing some of these findings with the team, it became apparent that the correct usage of the contracts is also guaranteed. An example of this also includes the usage of unchecked arithmetic, which is safe in its usage.

We would also like to note that lower-level code such as `createClone` was not thoroughly investigated.

Finally, the team chose to use its own pattern when it comes to updating contracts, using `AddressUpdatable`. It is not clear why this approach was chosen over simple `setters` as:

- Contracts' `_updateContractAdresses` do not perform any checks on the updated addresses
- Address can only be updated in groups (visible clearly in `DelegationAccountManager.sol` where adding a rewards manager requires the updating of all addresses). This approach seems to allow room for error.

Nevertheless, the structures and methods used throughout the project are well thought out and secure.

Regarding communication, the development team was very helpful in answering all our queries in a prompt manner.

DelegationAccountManager - Result of `createClone` is not checked

Finding ID: FYEO-FLARE-01
Severity: **Medium**
Status: **Remediated**

Description

When creating a new delegation account, the result of `createClone` is not addressed. This could lock an owner out of their delegation account as there is no way to delete a delegation account entry.

Proof of Issue

File name: contracts/personalDelegation/implementation/DelegationAccountManager.sol
Line number: 576

```
    function _getOrCreateDelegationAccountData() internal returns
(DelegationAccountData storage) {
        ...
          IIDelegationAccount delegationAccount =
IIDelegationAccount(payable(createClone(libraryAddress)));
    }
```

Severity and Impact Summary

Failing to create a new delegation account cannot be reverted.

Recommendation

Require that the clone was successfully created.

DelegationAccountManager - Updated WNat not reflected in DelegationAccountCloneable

Finding ID: FYEO-FLARE-02
Severity: **Low**
Status: **Remediated**

Description

Being an `AddressUpdatable` contract, the `DelegationAccountManager` can update some of its members like the address to the `wNat` token. Some of these variables are also forwarded to `DelegationAccountCloneable` accounts: - `wNat` is forwarded in: - `disableDelegationAccount` - `delegate` - `batchDelegate` - `undelegateAll` - `revokeDelegationAt` - `withdraw` - `transferExternalToken` - `governanceVP` is forwarded in - `delegateGovernance` - `undelegateGovernance`

An issue would arise in case the value of `wNat` changes during the contract's usage. This could desynchronize the tokens used by the delegation accounts, disabling functionality.

For example, `withdraw` would not work on an updated token without external transfers occurring.

Proof of Issue

File name: contracts/personalDelegation/implementation/DelegationAccountManager.sol
Line number: 595

```
    function _updateContractAddresses(
        bytes32[] memory _contractNameHashes,
        address[] memory _contractAddresses
    )
        internal override
    {
        ftsoManager = IFtsoManager(_getContractAddress(_contractNameHashes,
_contractAddresses, "FtsoManager"));
        wNat = WNat(payable(_getContractAddress(_contractNameHashes,
_contractAddresses, "WNat")));
        governanceVP = wNat.governanceVotePower();
```

Severity and Impact Summary

Changing the manager's token can disrupt delegation accounts.

Recommendation

If the token address is not meant to change, it should be passed on to the constructor and not updated in `_updateContractAddresses`.

Alternatively, restrictions can be placed in `_updateContractAddresses` to make sure that the address is not incorrectly updated.

A final option would be for the delegation accounts to hold the token address as a private variable, set during their construction.

DelegationAccountClone - Missing event in `transferExternalToken`

Finding ID: FYEO-FLARE-03
Severity: **Informational**
Status: **Remediated**

Description

No events are emitted during the delegation account's `transferExternalToken`.

Proof of Issue

File name: contracts/personalDelegation/implementation/DelegationAccountClonable.sol
Line number: 220

```
    function transferExternalToken(
        WNat _wNat,
        IERC20 _token,
        uint256 _amount
    ) external override onlyManager {
        require(
            address(_token) != address(_wNat),
            "Transfer from wNat not allowed"
        );
        bool success = _token.transfer(owner, _amount);
        require(success, ERR_TRANSFER_FAILURE);
    }
```

Severity and Impact Summary

Events can improve the traceability and auditability of the contracts.

Recommendation

Emit a `TransferExternalToken` event.

DelegationAccountManager - Missing events

Finding ID: FYEO-FLARE-05
Severity: **Informational**
Status: **Remediated**

Description

We suggest adding events for the following cases: - In `setMaxFeeValueWei` - In `setRegisterExecutorFeeValueWei` - Refunding excess amount in `setClaimExecutors`

Proof of Issue

File name: contracts/personalDelegation/implementation/DelegationAccountManager.sol
Line number: 394

```
    function setMaxFeeValueWei(uint256 _maxFeeValueWei) external override
onlyGovernance {
        require(_maxFeeValueWei > 0, ERR_VALUE_ZERO);
        maxFeeValueWei = _maxFeeValueWei; // XXXX missing emit
    }
```

Line number: 403

```
    function setRegisterExecutorFeeValueWei(uint256
_registerExecutorFeeValueWei) external override onlyGovernance {
        require(_registerExecutorFeeValueWei > 0, ERR_VALUE_ZERO);
        registerExecutorFeeValueWei = _registerExecutorFeeValueWei; // XXXX
missing emit
    }
```

Line number: 96

```
function setClaimExecutors(address[] memory _executors)
        external payable override nonReentrant
        returns (IDelegationAccount)
    {
        ...
        if (msg.value > totalExecutorsFee) {
            /* solhint-disable avoid-low-level-calls */
            //slither-disable-next-line arbitrary-send-eth
            (bool success, ) = msg.sender.call{value: msg.value -
totalExecutorsFee}(""); //nonReentrant
            /* solhint-enable avoid-low-level-calls */
            require(success, ERR_TRANSFER_FAILURE);
        }
        return delegationAccountData.delegationAccount;
    }
```

Severity and Impact Summary

Adding events can enhance the traceability of state changes.

Recommendation

Provide events for the cases above.

FtsoRewardManager - Missing events

Finding ID: FYEO-FLARE-08
Severity: **Informational**
Status: **Remediated**

Description

No events are emitted when activating or deactivating the contract.

Proof of Issue

File name: contracts/tokenPools/implementation/FtsoRewardManager.sol
Line number: 282

```
    function activate() external override onlyImmediateGovernance {
        require(inflation != address(0) && address(ftsoManager) != address(0)
&& address(wNat) != address(0),
            "addresses not set");
        active = true;
    }
```

Line number: 300

```
    function deactivate() external override onlyImmediateGovernance {
        active = false;
    }
```

Severity and Impact Summary

Activate events improve the traceability and auditability of the contract.

Recommendation

Emit activation events.

FtsoRewardManager - Optimization in `_isRewardClaimable`

Finding ID: FYEO-FLARE-09
Severity: **Informational**
Status: **Remediated**

Description

A boolean check in `_isRewardClaimable` can be optimized.

Proof of Issue

File name: contracts/tokenPools/implementation/FtsoRewardManager.sol
Line number: 1205

```
    function _isRewardClaimable(uint256 _rewardEpoch, uint256
_currentRewardEpoch) internal view returns (bool) {
        if (_rewardEpoch < nextRewardEpochToExpire ||
            _rewardEpoch >= _currentRewardEpoch ||
            _rewardEpoch < firstClaimableRewardEpoch) {
            // reward expired and closed or current or future or before
claiming enabled
            return false;
        }
        return true;
    }
```

Severity and Impact Summary

Extra checks infer some gas costs.

Recommendation

Simply

```
return _rewardEpoch >= firstClaimableRewardEpoch &&
       _rewardEpoch >= nextRewardEpochToExpire &&
       _rewardEpoch < _currentRewardEpoch
```

FtsoRewardManager - Unclear usage of `totalSelfDestructReceivedWei`

Finding ID: FYEO-FLARE-10
Severity: **Informational**
Status: **Remediated**

Description

FtsoRewardManager can receive Wei through `receiveInflation()`, which is the contract's only payable function. However, `_handleSelfDestructProceeds()` makes sure that excess balance is accounted for in `totalSelfDestructReceivedWei`. The amount stored in `totalSelfDestructReceivedWei` is never actually used, simply accounted for when checking `mustBalance`.

Proof of Issue

File name: contracts/tokenPools/implementation/FtsoRewardManager.sol
Line number: 798

```
    function _handleSelfDestructProceeds() internal returns (uint256
_currentBalance, uint256 _expectedBalance) {
        _expectedBalance = lastBalance.add(msg.value);
        _currentBalance = address(this).balance;
        if (_currentBalance > _expectedBalance) {
            // Then assume extra were self-destruct proceeds
            totalSelfDestructReceivedWei =
totalSelfDestructReceivedWei.add(_currentBalance).sub(_expectedBalance);
        } else if (_currentBalance < _expectedBalance) {
            // This is a coding error
            assert(false);
        }
    }
```

The only way for `totalSelfDestructReceivedWei` to be increased is if the contract has received Wei externally

Severity and Impact Summary

It is unclear whether this has a further impact.

Recommendation

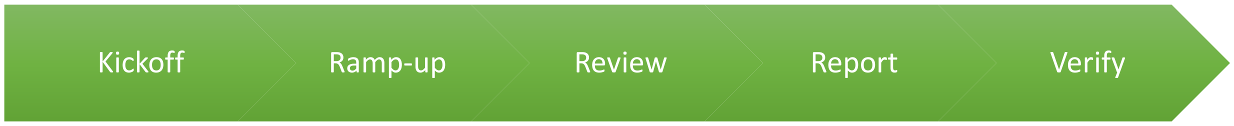
Use `receive` or `fallback` to control incoming amounts. This amount could be burned if it is intended to be not used.

OUR PROCESS

METHODOLOGY

FYEO Inc. uses the following high-level methodology when approaching engagements. They are broken up into the following phases.

Figure 2: Methodology Flow



KICKOFF

The project is kicked off as the sales process has concluded. We typically set up a kickoff meeting where project stakeholders are gathered to discuss the project as well as the responsibilities of participants. During this meeting we verify the scope of the engagement and discuss the project activities. It's an opportunity for both sides to ask questions and get to know each other. By the end of the kickoff there is an understanding of the following:

- Designated points of contact
- Communication methods and frequency
- Shared documentation
- Code and/or any other artifacts necessary for project success
- Follow-up meeting schedule, such as a technical walkthrough
- Understanding of timeline and duration

RAMP-UP

Ramp-up consists of the activities necessary to gain proficiency on the project. This can include the steps needed for familiarity with the codebase or technological innovation utilized. This may include, but is not limited to:

- Reviewing previous work in the area including academic papers
- Reviewing programming language constructs for specific languages
- Researching common flaws and recent technological advancements

REVIEW

The review phase is where most of the work on the engagement is completed. This is the phase where we analyze the project for flaws and issues that impact the security posture. Depending on the project this may include an analysis of the architecture, a review of the code, and a specification matching to match the architecture to the implemented code.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review of the original protocol
2. Review of the code written for the project
3. Compliance of the code with the provided technical documentation

The review for this project was performed using manual methods and utilizing the experience of the reviewer. No dynamic testing was performed, only the use of custom-built scripts and tools were used to assist the reviewer during the testing. We discuss our methodology in more detail in the following sections.

Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues
- Poor coding practices and unsafe behavior
- Leakage of secrets or other sensitive data through memory mismanagement
- Susceptibility to misuse and system errors
- Error management and logging

This list is general and not comprehensive, meant only to give an understanding of the issues we are looking for.

Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for things such as:

- Proper implementation of the documented protocol phases
- Proper error handling
- Adherence to the protocol logical description

Reporting

FYEO Inc. delivers a draft report that contains an executive summary, technical details, and observations about the project.

The executive summary contains an overview of the engagement including the number of findings as well as a statement about our general risk assessment of the project. We may conclude that the overall risk is low but depending on what was assessed we may conclude that more scrutiny of the project is needed.

We report security issues identified, as well as informational findings for improvement, categorized by the following labels:

- Critical
- High
- Medium
- Low
- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

As we perform the audit, we may identify issues that aren't security related, but are general best practices and steps that can be taken to lower the attack surface of the project. We will call those out as we encounter them and as time permits.

As an optional step, we can agree on the creation of a public report that can be shared and distributed with a larger audience.

VERIFY

After the preliminary findings have been delivered, this could be in the form of the approved communication channel or delivery of the draft report, we will verify any fixes within a window of time specified in the project. After the fixes have been verified, we will change the status of the finding in the report from open to remediated.

The output of this phase will be a final report with any mitigated findings noted.

ADDITIONAL NOTE

It is important to note that, although we did our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits along with the scope of the agreement.

While assessing the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. This is a solid baseline for severity determination.

THE CLASSIFICATION OF VULNERABILITIES

Security vulnerabilities and areas for improvement are weighted into one of several categories using, but is not limited to, the criteria listed below:

Critical – vulnerability will lead to a loss of protected assets

- This is a vulnerability that would lead to immediate loss of protected assets
- The complexity to exploit is low

- The probability of exploit is high

High - vulnerability has potential to lead to a loss of protected assets

- All discrepancies found where there is a security claim made in the documentation that cannot be found in the code
- All mismatches from the stated and actual functionality
- Unprotected key material
- Weak encryption of keys
- Badly generated key materials
- Txn signatures not verified
- Spending of funds through logic errors
- Calculation errors overflows and underflows

Medium - vulnerability hampers the uptime of the system or can lead to other problems

- Insecure calls to third party libraries
- Use of untested or nonstandard or non-peer-reviewed crypto functions
- Program crashes, leaves core dumps or writes sensitive data to log files

Low – vulnerability has a security impact but does not directly affect the protected assets

- Overly complex functions
- Unchecked return values from 3rd party libraries that could alter the execution flow

Informational

- General recommendations