F Y E O

Secure Code Review of the Flare Validator V2

Flare Networks Ltd.

October 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

EXECUTIVE SUMMARY

OVERVIEW

Flare Networks Ltd. engaged FYEO Inc. to perform a second Secure Code Review of the Flare Validator.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on September 26 - September 30, 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 and work performed by the FYEO Security Team.

KEY FINDINGS

During the review, no vulnerabilities or major security concerns were identified.

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 the Flare Validator V2. 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://github.com/sprwn/go-flare with the commit hash `a60fac5ab26335b5761a5ba6e7b4ee51de2d1dbf`.

At the start of the review, the Flare team migrated their code to a public repository at https://github.com/flare-foundation/go-flare with the commit hash `0b9ebef794a8cfe68c04d1da5e284a2a94fb8a2c`. No code changes had been made upon the migration.

The scope of this review entailed the developments made since the previous review which was supplied through https://github.com/sprwn/go-flare with the commit hash `b596931f85367ca5db272e962b2fded14a9733f1`.

CODE REVIEW SUMMARY

GENERAL OBSERVATIONS

The Flare Network is a Layer-1 blockchain aiming to connect everything, including off-chain data and other Layer-1 blockchains through their F-Asset, State Connector and Layer Cake protocols.

Code reviewed

This audit is focused on the validator golang codebase of the Flare Network. The FYEO Security Team reviewed all changes from commit `b596931f85367ca5db272e962b2fded14a9733f1` to commit `a60fac5ab26335b5761a5ba6e7b4ee51de2d1dbf` (aka `0b9ebef794a8cfe68c04d1da5e284a2a94fb8a2c` from the public repository) and there were no vulnerabilities or major security concerns found during this review.

The major changes since the last review include:

1. A patch to avalanchego related to a flaw in their `NativeAssetCall` contract

   (commits `b596931f85367ca5db272e962b2fded14a9733f1`, `289ab626e12fbbd6e779721f0d75211b40211bfc`, and `9b5f83195930f2fdcb709da03b1721b0eedcfa25`)

   - The FYEO team checked the vulnerability from `NativeAssetCall`. `NativeAssetCall` is a precompile contract that transfers Avalanche Native Token (if the receiver is not a contract) or invokes a contract (if the receiver is a contract). The vulnerability is from the ability to callback the `NativeAssetCall` precompile contract.

   - The Flare team elected to stay on the stable v1.7.18 version of avalanchego (commit `e946ff1f5e97d1ae4b3399a34c17667d40130be0`) and disable the `NativeAssetCall` functionality so that the call will return an error if executed after September 16th at 15:00 UTC. The patch looks good, and the gas was also subtracted.

2. Coston network is updated to Costwo network (commit `e9219c601717d4194b0f5192d61cf19db07c68f9`). All costwo network changes are updated accordingly.
3. Governance setting is added to the golang level (commit `d7e3ee4d7e0c32d65aa55583bf28b5baaf5f300f`). New changes are added to `state_transition.go` and a new file `governance_setting.go` is added.
   - State transition is where the privileged call to governance setting is triggered. The FYEO team reviewed changes added to the state transition. All necessary checks are performed:
     - The call to set governance address or time lock is only triggered if a transaction that invokes the right governance setting contract is activated (https://gitlab.com/flarenetwork/flare-smart-contracts/-

/blob/380323c5b6868e5d622460aa49c2e796b31e7d81/contracts/genesis/implementation/GovernanceSettings.sol)
        - To the right functions (set governance address or time lock)
        - With the original Coinbase `0x0100000000000000000000000000000000000000`
        - And length of the message data is 36.
    - `governance_setting.go` code will trigger the privileged call by updating the original Coinbase with Coinbase Signal only if the call data matches the hardcoded governance address or time lock. These values can be updated via hard forks. This is a secure and stable way to ensure that new off-chain data is inputted to the blockchain safely.

Summary

Code quality is very good; operations are carried out carefully. The Flare development team was very communicative, quickly providing responses to the auditing team.

Our Process

Methodology

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

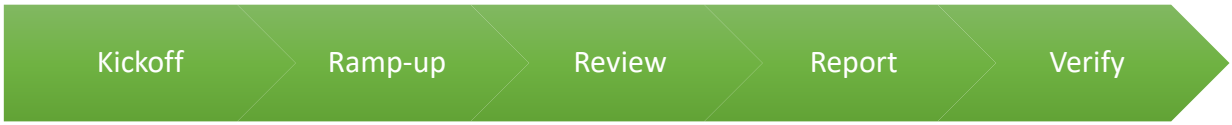

Figure 1: Methodology Flow

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