FYEO

Security Code Review of Real-time Dashboard

September 2025
Version 1.0

Presented by:
FYEO Inc.
PO Box 147044
Lakewood CO 80214
United States

Security Level
Public

Table of Contents

Executive Summary

Overview

Real-time Dashboard engaged FYEO Inc. to perform a Security Code Review.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on August 25 - August 29, 2025, 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 have been identified during the testing period. These should be prioritized for remediation to reduce the risk they pose:

- FYEO-REAL-01 – Incomplete Vote Tallying Logic
- FYEO-REAL-02 – Code clarity

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 Security Code Review of Real-time Dashboard. 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 public repository at https://github.com/3uild-3thos/govcontract/ with the commit hash 4db2a43cc5df76ffa2c285cf045f8d455c95b809.

Remediations were submitted with the commit hash 522f1c79a7cee002921adf5ddca8555290c829d6.

Files included in the code review
```
govcontract/
└── contract/
    └── programs/
        └── govcontract/
            └── src/
                ├── instructions/
                │   ├── cast_vote.rs
                │   ├── create_proposal.rs
                │   ├── initialize_index.rs
                │   ├── mod.rs
                │   ├── modify_vote.rs
                │   ├── support_proposal.rs
                │   └── tally_votes.rs
                ├── state/
                │   ├── mod.rs
                │   ├── proposal.rs
                │   ├── proposal_index.rs
                │   ├── support.rs
                │   └── vote.rs
                ├── error.rs
                ├── lib.rs
                └── utils.rs
```

```
govcontract/
└── contract/
    └── programs/
        └── govcontract/
            └── src/
                ├── instructions/
                │   ├── cast_vote.rs
                │   ├── create_proposal.rs
                │   ├── initialize_index.rs
                │   ├── mod.rs
                │   ├── modify_vote.rs
                │   ├── support_proposal.rs
                │   └── tally_votes.rs
                ├── state/
                │   ├── mod.rs
                │   ├── proposal.rs
                │   ├── proposal_index.rs
                │   ├── support.rs
                │   └── vote.rs
                ├── error.rs
                ├── lib.rs
                └── utils.rs
```

Table 1: Scope

Technical Analyses and Findings

During the Security Code Review of Real-time Dashboard, we discovered:

- 1 finding with MEDIUM severity rating.
- 1 finding with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

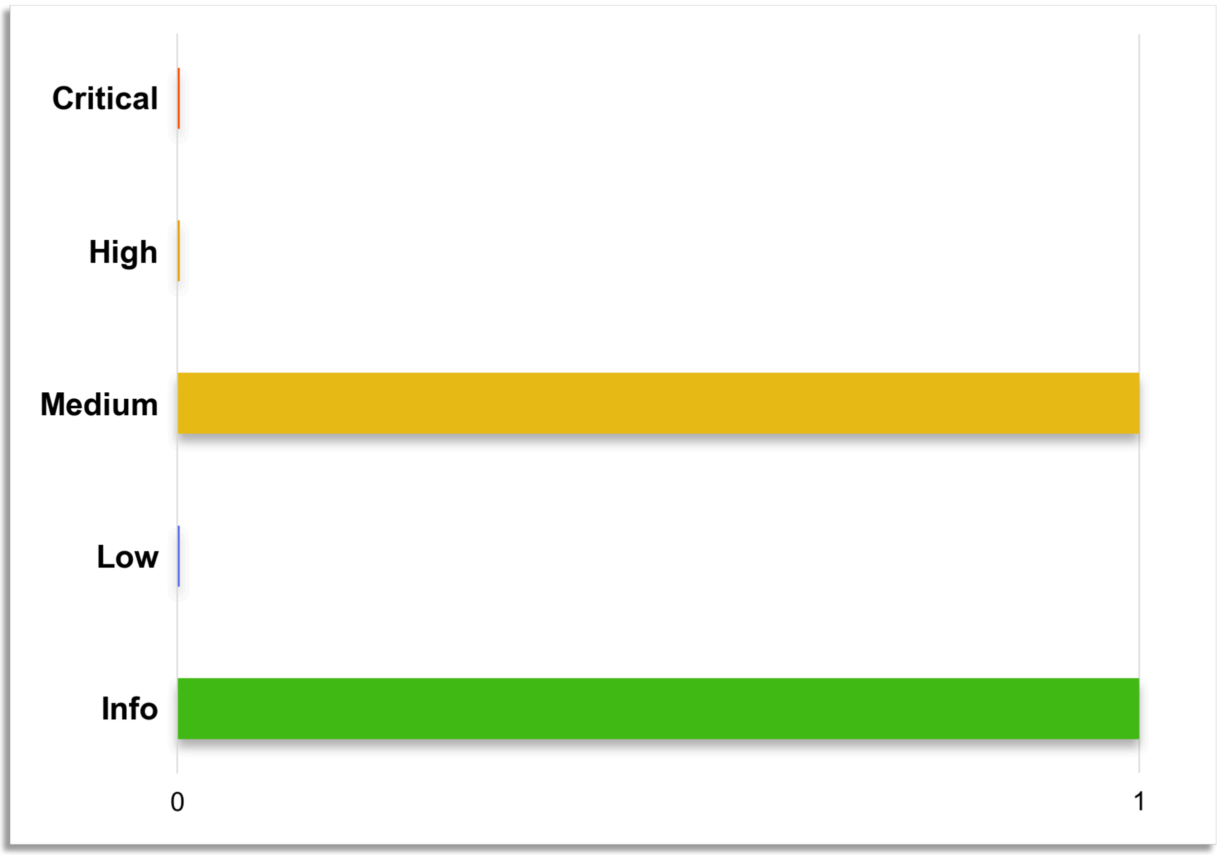

Figure 1: Findings by Severity

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	Status
FYEO-REAL-01	Medium	Incomplete Vote Tallying Logic	Remediated
FYEO-REAL-02	Informational	Code clarity	Remediated

Table 2: Findings Overview

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

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.

Conclusion

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

Technical Findings

General Observations

The Solana Validator Governance System is a program that allows Solana validators to participate in governance through proposal creation, voting, and result tallying. At its core, the program defines data structures for proposals and votes, enforces rules for proposal lifecycles, and enables stake-weighted decision-making. Validators can create proposals with a title and description (Github link), signal support to bring them to a voting phase, and cast votes with configurable weights for “For,” “Against,” or “Abstain.” Each vote is tied to a validator’s vote account, and the validator’s stake determines the weight of their voting power. Once a proposal’s voting period ends, the program processes votes by reading validator stake amounts and aggregating them to determine the overall outcome.

The system is composed of three integrated components that work together to make governance accessible and transparent. The smart contract, written in Rust using the Anchor framework, implements all logic for proposals, voting, and finalization directly on the Solana blockchain. A command-line interface (CLI) called svmgov allows validators to interact with the contract by creating proposals, supporting them, casting votes, and triggering tally operations through transactions. For a broader audience, a web-based frontend provides a read-only interface to view proposals, track their status, and monitor vote distributions without requiring technical expertise. Together, these components provide a stake-weighted governance system that leverages Solana’s high throughput and low-latency infrastructure to manage validator-driven decision-making at scale.

Incomplete Vote Tallying Logic

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

Description

The `TallyVotes` instruction contains several issues that may lead to unexpected panics and potential DoS situations. Key concerns include: * Any validator can initiate tallying for a proposal - it is unclear if this is desired behaviour * Vote accounts are de-duplicated using a realloc(0), which leaves 0 data accounts with unreclaimable lamports * If VoteState accounts are closed between epochs, the tally function will panic * The vote count is decremented in a loop without underflow protection, which could panic if extra accounts are supplied (More than vote_count*2 accounts could be supplied)* Iterating over all votes in one transaction may exceed compute limits on large proposals. There is also a transaction size limit to consider which limits the number of accounts that can realistically be passed into the instruction

Proof of Issue

File name: contract/programs/govcontract/src/instructions/tally_votes.rs
Line number: 174, various others

```
vote.to_account_info().realloc(0, false)?;

vote_count -= 1;
```

Severity and Impact Summary

The code and logic appear unfinished and may not work as intended. Potentially preventing the program from serving its purpose.

Recommendation

1. Introduce explicit access control or document intended behaviour.
2. Handle a potentially closed VoteState account gracefully.
3. Handle additional accounts gracefully (vote_count*2).
4. Use a `tallied` flag or archive mechanism instead of `realloc(0, false)`.
5. Consider batching tally operations or introducing pagination for proposals with large voter sets. Use the tallied flag.
6. Provide a refund mechanism after the tally is complete.

Code clarity

Finding ID: FYEO-REAL-02
Severity: Informational
Status: Remediated

Description

The codebase uses hard-coded numeric values (magic numbers), repeated validation patterns, and redundant `Result` wrapping, all of which reduce readability, consistency, and maintainability.

Proof of Issue

File name: contract/programs/govcontract/src/state/proposal.rs
Line number: 7

```
#[max_len(50)]
pub title: String,

#[max_len(250)]
pub description: String,
```

File name: contract/programs/govcontract/src/instructions/create_proposal.rs
Line number: 54

```
require!(title.len() <= 50, GovernanceError::TitleTooLong);
require!(description.len() <= 250, GovernanceError::DescriptionTooLong);
```

File name: contract/programs/govcontract/src/utils.rs
File name: contract/programs/govcontract/src/instructions/cast_vote.rs
File name: contract/programs/govcontract/src/instructions/modify_vote.rs
File name: contract/programs/govcontract/src/instructions/tally_votes.rs

```
10_000
```

File name: various

```
let vote_account_data = self.spl_vote_account.data.borrow();
let version = u32::from_le_bytes(
    vote_account_data[0..4]
        .try_into()
        .map_err(|_| GovernanceError::InvalidVoteAccount)?,
);
require!(version <= 2, GovernanceError::InvalidVoteAccount);


let node_pubkey =
    Pubkey::try_from(&vote_account_data[4..36]).map_err(|_|
GovernanceError::InvalidVoteAccount)?;

require_keys_eq!(
    node_pubkey,
    self.signer.key(),
    GovernanceError::InvalidVoteAccount
);
```

This check is used in many places and duplicated for each instance.

File name: contract/programs/govcontract/src/lib.rs

```
ctx.accounts.support_proposal(&ctx.bumps)?;
Ok(())
```

All handlers use this pattern, but could just return the handler call.

Severity and Impact Summary

These issues are not security concerns but will degrade the maintainability of the code base.

Recommendation

1. Replace magic numbers with named constants.
2. Abstract `VoteAccount` parsing.
3. Simplify redundant return patterns

Our Process

Methodology

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

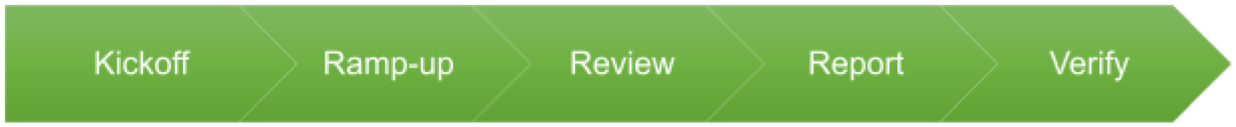

Figure 2: Methodology Flow

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.