F Y E O

Security Code Review of Solana Staker Vote Override

Exo Tech

November 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

Exo Tech engaged FYEO Inc. to perform a Security Code Review of Solana Staker Vote Override.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on October 06 - October 10, 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-EXO-01 – Add operators may introduce duplicates and exceed operator limit
- FYEO-EXO-02 – Pushes in cast_vote may exceed limits
- FYEO-EXO-03 – Changing the operator whitelist mid-ballot affects consensus calculations
- FYEO-EXO-04 – RemoveVote leaves zeroed BallotTally entries and can cause unbounded structure growth
- FYEO-EXO-05 – Service concerns
- FYEO-EXO-06 – SetTieBreaker can set a winning ballot even though the operator set changed or tally is zero
- FYEO-EXO-07 – Unbounded gzip decompression in read_from_bytes_with_hash
- FYEO-EXO-08 – Unnecessary clone allocation

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 Solana Staker Vote Override. 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/exo-tech-xyz/gov-v1 with the commit hash 8e31a6fcb8e40cd4b521018f8748c0a5f0c2ffce.

Remediations were provided with the commit hash bc3c78173df6ea1953beead06802514f013b7cc9.

Files included in the code review

```
gov-v1/
├── cli/
│   └── src/
│       ├── utils/
│       │   ├── mod.rs
│       │   ├── parsers.rs
│       │   └── send_utils.rs
│       ├── consts.rs
│       ├── lib.rs
│       ├── main.rs
│       └── merkle.rs
├── programs/
│   └── gov-v1/
│       └── src/
│           ├── instructions/
│           │   ├── ballot/
│           │   │   ├── cast_vote.rs
│           │   │   ├── finalize_ballot.rs
│           │   │   ├── init_ballot_box.rs
│           │   │   ├── mod.rs
│           │   │   ├── remove_vote.rs
│           │   │   └── set_tie_breaker.rs
│           │   ├── program_config/
│           │   │   ├── init_program_config.rs
│           │   │   ├── mod.rs
│           │   │   ├── update_operator_whitelist.rs
│           │   │   └── update_program_config.rs
│           │   ├── verify/
│           │   │   ├── close_meta_merkle_proof.rs
│           │   │   ├── init_meta_merkle_proof.rs
│           │   │   ├── mod.rs
│           │   │   └── verify_merkle_proof.rs
```

Files included in the code review

```
│           │   └── mod.rs
│           ├── state/
│           │   ├── ballot_box.rs
│           │   ├── consensus_result.rs
│           │   ├── mod.rs
│           │   ├── program_config.rs
│           │   └── proof.rs
│           ├── error.rs
│           ├── lib.rs
│           └── merkle_helper.rs
├── tests/
│   └── src/
│       ├── utils/
│       │   ├── assert.rs
│       │   ├── data_types.rs
│       │   ├── fetch_utils.rs
│       │   └── mod.rs
│       ├── lib.rs
│       └── test_full_program_flow.rs
└── verifier-service/
    ├── src/
    │   ├── bin/
    │   │   └── loadtest.rs
    │   ├── database/
    │   │   ├── models/
    │   │   │   ├── database.rs
    │   │   │   ├── mod.rs
    │   │   │   └── views.rs
    │   │   ├── constants.rs
    │   │   ├── migrator.rs
    │   │   ├── mod.rs
    │   │   ├── operations.rs
    │   │   └── sql.rs
    │   ├── main.rs
    │   ├── metrics.rs
    │   ├── middleware.rs
    │   ├── types.rs
    │   ├── upload.rs
    │   └── utils.rs
    └── tests/
        ├── common/
        │   └── mod.rs
        └── e2e_binary.rs
```

Table 1: Scope

Technical Analyses and Findings

During the Security Code Review of Solana Staker Vote Override, we discovered:

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

The following chart displays the findings by severity.

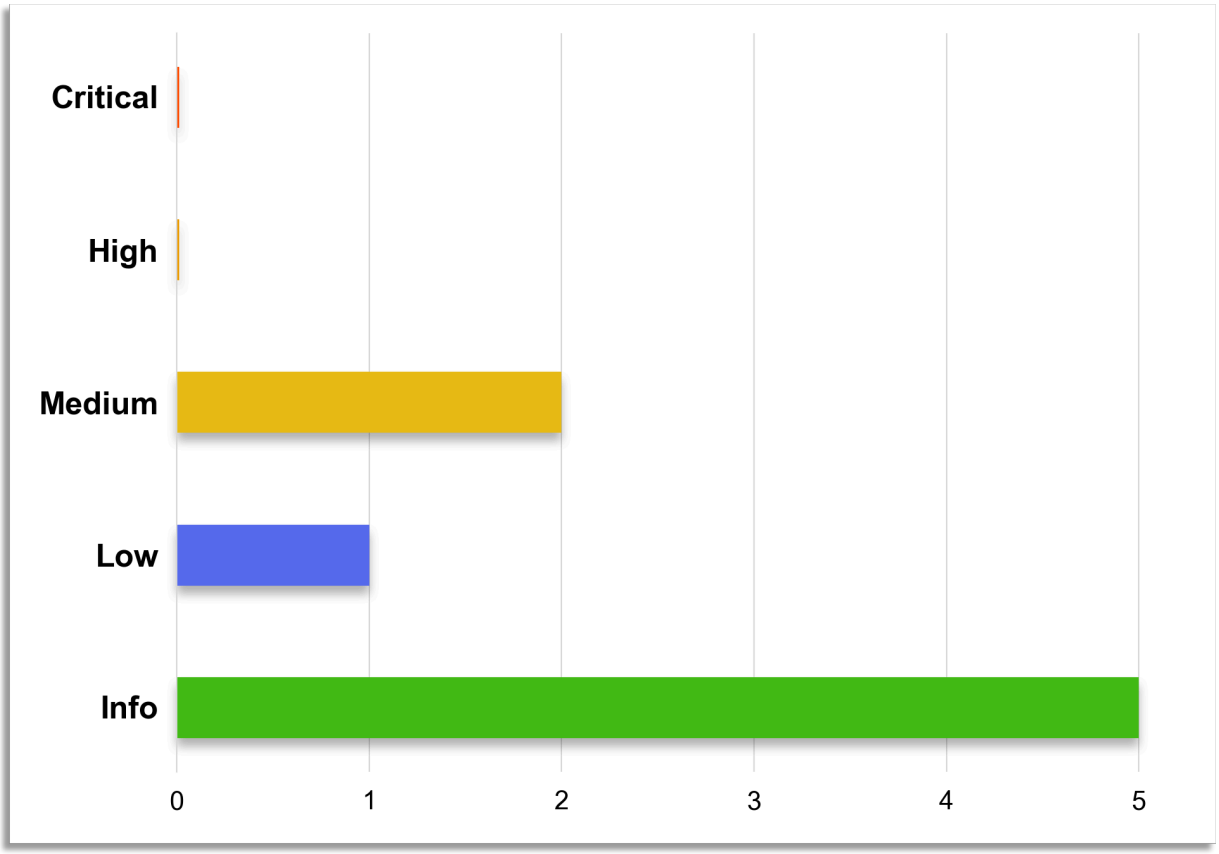

Figure 1: Findings by Severity

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	Status
FYEO-EXO-01	Medium	Add operators may introduce duplicates and exceed operator limit	Remediated
FYEO-EXO-02	Medium	Pushes in cast_vote may exceed limits	Remediated
FYEO-EXO-03	Low	Changing the operator whitelist mid-ballot affects consensus calculations	Acknowledged
FYEO-EXO-04	Informational	RemoveVote leaves zeroed BallotTally entries and can cause unbounded structure growth	Acknowledged
FYEO-EXO-05	Informational	Service concerns	Remediated
FYEO-EXO-06	Informational	SetTieBreaker can set a winning ballot even though the operator set changed or tally is zero	Remediated
FYEO-EXO-07	Informational	Unbounded gzip decompression in read_from_bytes_with_hash	Remediated
FYEO-EXO-08	Informational	Unnecessary clone allocation	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 program is a governance snapshot voting system that helps a small group of trusted operators agree on which set of validator stake data should be used for on-chain governance decisions. Operators review and vote on candidate snapshots (files describing how stake is distributed across validators), and when a clear majority is reached the chosen snapshot is recorded as the authoritative source for subsequent verification and voting. The system also provides a way to break ties or resolve deadlocks when voting fails to reach consensus by a deadline, ensuring the process can always move forward.

In addition to coordinating the group decision, the program stores compact proofs that let anyone verify an individual validator's stake without re-uploading files. That makes it practical to confirm a validator's voting weight during governance actions while keeping on-chain storage and transaction sizes small. Overall, it's designed to make multi-operator agreement on stake snapshots safe, auditable, and reusable so the wider governance process can rely on a single agreed-upon snapshot.

Add operators may introduce duplicates and exceed operator limit

Finding ID: FYEO-EXO-01
Severity: Medium
Status: Remediated

Description

`add_operators` builds a HashSet of existing operators but does **not** de-duplicate the `operators_to_add` input; duplicate entries in the input will still be pushed for each occurrence. No capacity check is done before pushing, so a malicious/buggy input could overflow expected operator limits.

Proof of Issue

File name: programs/gov-v1/src/state/program_config.rs
Line number: 46

```
pub fn add_operators(&mut self, operators_to_add: Option<Vec<Pubkey>>) {
    if let Some(operators) = operators_to_add {
        let existing: HashSet<Pubkey> =
self.whitelisted_operators.iter().cloned().collect();
        for op in operators {
            if !existing.contains(&op) {
                self.whitelisted_operators.push(op);
            }
        }
    }
}
```

Severity and Impact Summary

Duplicate entries increase `whitelisted_operators` length, can break consensus math or allow exceeding the intended max length.

Recommendation

Deduplicate `operators_to_add` before iterating. Enforce a hard cap on the final result.

Pushes in cast_vote may exceed limits

Finding ID: FYEO-EXO-02
Severity: Medium
Status: Remediated

Description

When creating a new ballot tally or operator vote the handler unconditionally pushes to `ballot_box.ballot_tallies` and `ballot_box.operator_votes`. If operators can be added and removed concurrently, these pushes can grow beyond expected safe bounds, causing panics, account size issues, or consensus math breakage.

Proof of Issue

File name: programs/gov-v1/src/instructions/ballot/cast_vote.rs
Line number: 56, 65

```
// If ballot is new, create a new BallotTally.
if !found {
    let new_ballot_tally = BallotTally {
        index: ballot_box.ballot_tallies.len().try_into().unwrap(),
        ballot: ballot.clone(),
        tally: 1,
    };
    tally = 1;
    ballot_index = new_ballot_tally.index;
    ballot_box.ballot_tallies.push(new_ballot_tally);// <- unguarded push
}

// ...
ballot_box.operator_votes.push(new_operator_vote);// <- unguarded push
```

Severity and Impact Summary

Unchecked growth may hit program/account limits, cause panics, allow more operator votes than designed, or break on-chain sizing invariants.

Recommendation

Add explicit bounds checks before pushing. Enforce `MAX_OPERATORS` and `MAX_TALLIES` invariants and return meaningful errors instead of panicking.

Changing the operator whitelist mid-ballot affects consensus calculations

Finding ID: FYEO-EXO-03
Severity: Low
Status: Acknowledged

Description

Ballot consensus uses `program_config.whitelisted_operators.len()` at vote time to compute `tally_bps`. That means changing the whitelist while a ballot is live changes the denominator used to compute consensus thresholds. The README states thresholds are configured on the `BallotBox` at creation, but the code does not snapshot the operator list at ballot creation.

Proof of Issue

File name: programs/gov-v1/src/instructions/ballot/cast_vote.rs
Line number: 68

```
if !ballot_box.has_consensus_reached() {
    let tally_bps =
        u64::from(tally) * 10000 / (program_config.whitelisted_operators.len() as u64);
    if tally_bps >= ballot_box.min_consensus_threshold_bps.into() {
        ballot_box.slot_consensus_reached = clock.slot;
        ballot_box.winning_ballot = ballot;
    }
}
```

Severity and Impact Summary

Changing the whitelist during voting can:
* Increasing denominator makes it impossible for an otherwise-leading ballot to reach consensus. * Decreasing denominator retroactively makes a ballot reach consensus.

Recommendation

Snapshot the operator set into `BallotBox` at creation and use that snapshot for consensus math, or always recalculate consensus based on a ballot-specific operator snapshot.
Update README to reflect the actual design and document whether operator lists are ballot-scoped or global.

RemoveVote leaves zeroed BallotTally entries and can cause unbounded structure growth

Finding ID: FYEO-EXO-04
Severity: **Informational**
Status: **Acknowledged**

Description

Removing an operator's vote decrements the `BallotTally.tally` but leaves the `BallotTally` entry even when `tally == 0` to preserve ordering. Repeated vote/remove cycles across different ballots can result in many zero-tally entries, consuming storage/slots and eventually hitting limits.

Proof of Issue

File name: programs/gov-v1/src/instructions/ballot/remove_vote.rs
Line number: 44

```
// Decrement tally on BallotTally. BallotTally is kept even when tally is 0 to
maintain
// order of indices.
let ballot_tally = &mut ballot_box.ballot_tallies[ballot_index as usize];
ballot_tally.tally = ballot_tally.tally.checked_sub(1).unwrap();
```

Severity and Impact Summary

While it was said that all operators are trusted, a malicious actor can cause many zero-tally rows (tombstones) by repeatedly voting and removing, consuming the limited on-chain account space.

Recommendation

Consider not allowing repeated votes for one operator. Or consider re-indexing.

Service concerns

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

Description

Because the service is typically run by a trusted operator, the practical risk is reduced, but these issues still enable accidental data loss, partial/invalid indexing, operator errors, spoofable rate-limit bypasses, and local file-system manipulation.

Proof of Issue

1) **`database::init_pool`** - arbitrary DB path and file creation, may overwrite system files.

```
   pub async fn init_pool(db_path: &str) -> Result<SqlitePool> {
       info!("Opening database at {:?}", db_path);

       // Ensure parent directory exists
       if db_path != ":memory:" {
           let path = Path::new(db_path);
           if let Some(parent) = path.parent() {
               if !parent.as_os_str().is_empty() {
                   fs::create_dir_all(parent)?;
               }
           }
           // Create the DB file if it doesn't exist
           if !path.exists() {
               fs::File::create(path)?;
           }
       }
       // ...
   }
```

2) **`VoteAccountRecord::insert`** - `INSERT OR REPLACE` can overwrite/preserve unintended fields (e.g. `created_at`), causing data loss or losing metadata. Prefer `INSERT ... ON CONFLICT(key) DO UPDATE`

```
   sqlx::query(
       "INSERT OR REPLACE INTO vote_accounts
        (network, snapshot_slot, vote_account, voting_wallet, stake_merkle_root,
active_stake, meta_merkle_proof)
        VALUES (?, ?, ?, ?, ?, ?, ?)",
   )
   .bind(&self.network)
   // ...
```

3) **`inject_client_ip` middleware** - header trust without provenance/whitelist makes X-Forwarded-For / CF-Connecting-IP spoofable and may allow bypassing IP-based rate-limiting.

```
   let cf_name: HeaderName = HeaderName::from_static("cf-connecting-ip");
   let xff_name: HeaderName = HeaderName::from_static("x-forwarded-for");
```

4) **`index_snapshot_data`** - per-row inserts (no transaction) while indexing snapshot; partial failures can leave the DB in a partially indexed state.

```
snapshot_meta.insert(pool).await?;
// ...
vote_account_record.insert(pool).await?;
// ...
stake_account_record.insert(pool).await?;
```

Severity and Impact Summary

Impacts: * **Arbitrary `db_path`:** allows accidental creation of files anywhere the process can write; can lead to surprising behavior, privilege escalation surface. May also allow symlink/traversal issues if untrusted input used. * **`INSERT OR REPLACE`:** silently replaces entire row, possibly resetting fields (timestamps, metadata), causing data integrity loss and confusing state after re-indexing or updates. * **Header trust (`X-Forwarded-For`/`CF-Connecting-IP`):** unauthenticated clients can spoof headers to evade per-IP rate limits or cause incorrect telemetry/attribution. * **Non-transactional indexing:** crashes or partial failures during a snapshot import leave the DB partially populated (some vote/stake rows present, others missing), causing inconsistent query results and difficult recovery.

Recommendation

1. Avoid implicitly creating files in arbitrary paths from user-supplied env vars; fail early if `db_path` looks suspicious.
2. Use explicit UPSERT semantics instead of `INSERT OR REPLACE`.
3. Do not rely solely on header derived IP for security critical rate limiting.
4. Use a DB transaction around `snapshot_meta.insert(...)` so any failure rolls back and the DB stays consistent.

SetTieBreaker can set a winning ballot even though the operator set changed or tally is zero

Finding ID: FYEO-EXO-06
Severity: **Informational**
Status: **Remediated**

Description

`SetTieBreaker` simply assigns `slot_consensus_reached` and copies a ballot from input without validating current operator set, current tallies, or whether the chosen tally is non-zero. Essentially the Tie-Breaker can set whatever they want.

Proof of Issue

File name: programs/gov-v1/src/instructions/ballot/set_tie_breaker.rs
Line number: 27

```
ballot_box.slot_consensus_reached = clock.slot;
ballot_box.winning_ballot = ballot_box.ballot_tallies[ballot_index as usize]
    .ballot
    .clone();
```

Severity and Impact Summary

The tie-breaker may select a ballot with zero votes, or conflict with late consensus if operator set changed.

Recommendation

Validate that the chosen `ballot_index` exists and that `ballot_tallies[ballot_index].tally > 0`. Recompute/verify consensus using a snapshot/immutable operator set or reject tie-break if consensus became true between expiry and tie-break.

Unbounded gzip decompression in read_from_bytes_with_hash

Finding ID: FYEO-EXO-07
Severity: **Informational**
Status: **Remediated**

Description

`read_from_bytes_with_hash` performs gzip/deflate decompression without limiting the total decompressed size. A crafted compressed payload can expand to arbitrarily large size (zip bomb), leading to memory exhaustion / DoS.

Proof of Issue

File name: cli/src/merkle.rs
Line number: 31

```rust
pub fn read_from_bytes_with_hash(
    buf: Vec<u8>,
    is_compressed: bool,
) -> io::Result<(Self, Hash)> {
    let decompressed_buf = if is_compressed {
        let mut decoder = GzDecoder::new(&buf[..]);
        let mut decompressed = Vec::new();
        decoder.read_to_end(&mut decompressed)?;
        decompressed
    } else {
        buf
    };
```

Severity and Impact Summary

While this is a CLI tool, forged data could cause the program to allocate huge memory and crash.

Recommendation

Validate format/headers before decompression where possible. Safely abort with some limits.

Unnecessary clone allocation

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

Description

Code clones a possibly large `meta_merkle_proof` structure where a borrow/slice would suffice. This wastes CUs.

Proof of Issue

File name: programs/gov-v1/src/instructions/verify/verify_merkle_proof.rs
Line number: 41

```
verify_helper(
    leaf_content,
    meta_merkle_proof.meta_merkle_proof.clone(),
    consensus_result.ballot.meta_merkle_root.into(),
)?;
```

Severity and Impact Summary

Extra allocations; minor performance and memory overhead. Not functionally incorrect.

Recommendation

Replace `.clone()` with a borrow/reference.

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.