F Y E O

Security Code Review of Bio Launchpad

Bio

March 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

Bio engaged FYEO Inc. to perform a Security Code Review of the Bio Launchpad.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on February 11 - February 20, 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-BIO-01 – Possibility of a DOS attack
- FYEO-BIO-02 – Missing bound checks
- FYEO-BIO-03 – Tokens can only be withdrawn to ATA
- FYEO-BIO-04 – Transfer can be done with 0 amount
- FYEO-BIO-05 – Unfinished code

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 Bio Launchpad. 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/ASCorreia/bio-launchpad with the commit hash df250494e70fd3ab36ee4ee8729ff612c5a661ad.

Remediations were submitted with the commit hash f196db25fde03e716bfc9c313a328f5cb8bee06b. This includes additional updates unrelated to the issues identified during the audit. The review of the remediations does not include a review of the new functionality added.

Files included in the code review

```
bio-launchpad/
└── programs/
    └── bio-launchpad/
        └── src/
            ├── events/
            │   ├── curation.rs
            │   ├── curator.rs
            │   ├── dao.rs
            │   └── mod.rs
            ├── instructions/
            │   ├── close_curation.rs
            │   ├── curation.rs
            │   ├── init_curation.rs
            │   ├── init_dao.rs
            │   ├── mod.rs
            │   ├── start_bonding.rs
            │   └── withdrawal_curation.rs
            ├── state/
            │   ├── bonding.rs
            │   ├── curation.rs
            │   ├── curator.rs
            │   ├── dao_accounts.rs
            │   └── mod.rs
            ├── constants.rs
            ├── errors.rs
            └── lib.rs
```

Table 1: Scope

Technical Analyses and Findings

During the Security Code Review of Bio Launchpad, we discovered:

- 1 finding with LOW severity rating.
- 4 findings with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

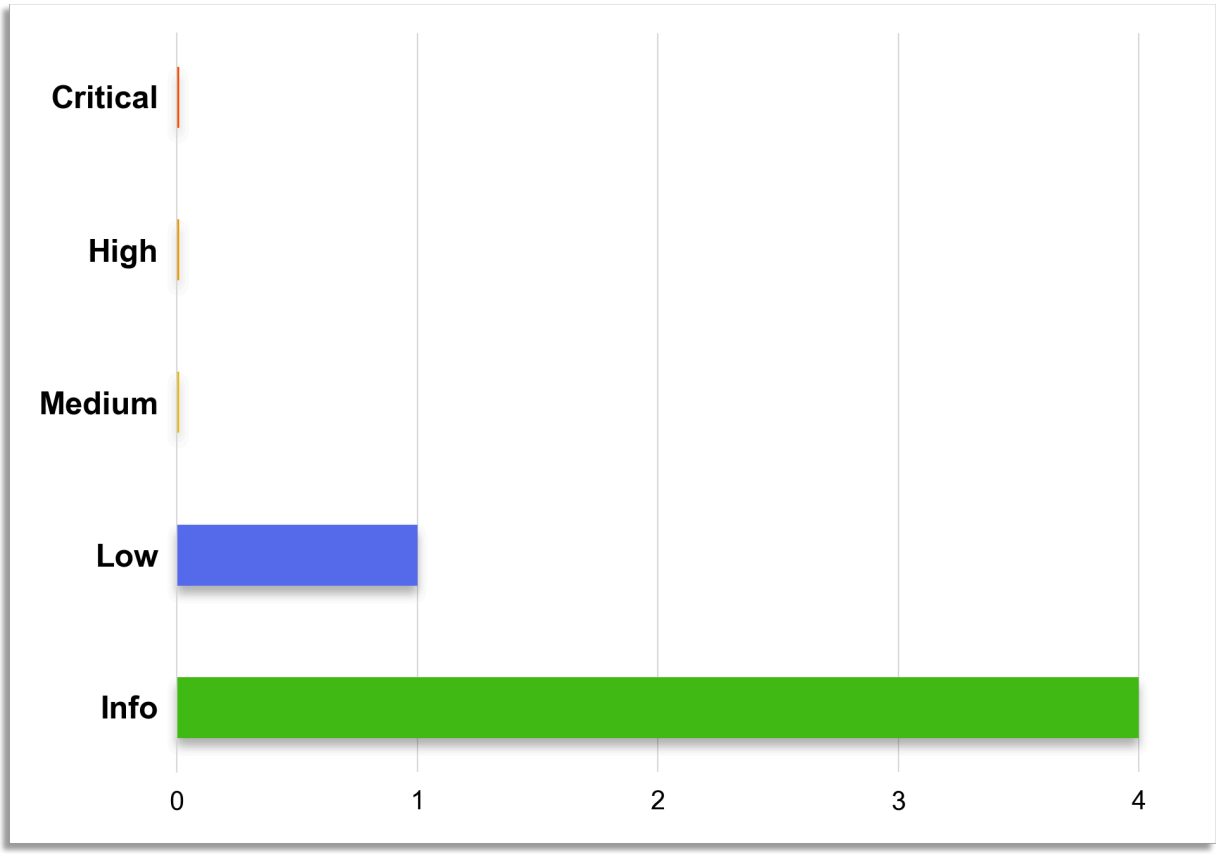

Figure 1: Findings by Severity

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-BIO-01	Low	Possibility of a DOS attack
FYEO-BIO-02	Informational	Missing bound checks
FYEO-BIO-03	Informational	Tokens can only be withdrawn to ATA
FYEO-BIO-04	Informational	Transfer can be done with 0 amount
FYEO-BIO-05	Informational	Unfinished code

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.

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 Bio Launchpad is a decentralized platform built on Solana to support the creation, funding, and governance of biotech-focused DAOs. It enables scientific communities, patients, and biotech professionals to collectively fund and develop tokenized biotech projects. Utilizing the BIO token, the platform incorporates a governance mechanism where token holders participate in curating and selecting new BioDAOs through staking, ensuring long-term alignment with the ecosystem.

The review of this Solana program showed that the code is well-structured and thoughtfully developed. It makes effective use of the Anchor framework and follows a clear, organized design. The team has been highly responsive and communicative throughout the review process, contributing to an efficient and collaborative experience.

Possibility of a DOS attack

Finding ID: FYEO-BIO-01
Severity: **Low**
Status: **Remediated**

Description

There is a minor DOS opportunity as the current minimum contribution could be 1 token unit (for a token worth 1 USD that could be a millionth of 1 USD depending on decimals configured). There are only `u16`, so about ~65k contributors possible. This means someone could potentially fill all slots for as little as a few USD. The transaction costs may be the most expensive part of this as it currently stands.

Proof of Issue

File name: programs/bio-launchpad/src/instructions/curation.rs
Line number: 87

```
self.curation_account.total_curators =
self.curation_account.total_curators.checked_add(1).ok_or(BioError::ContributionOverflow
)?;
```

This is a `u16`: `pub total_curators: u16,`. And the minimum contribution is currently 1 token.

Severity and Impact Summary

With the current setup of 1 token contribution and a limit of ~65k contributors, the contract could be blocked with worthless contributions. With ~65k contributors contributing the minimum, the contract would raise 0.065 USD with a 6 decimal token or 0.000065 USD for a 9 decimal token (assuming 1 token is priced around 1 USD).

Recommendation

The minimum contribution should be calculated such that the funding goal can be reached even if all contributions are the minimum value.

Missing bound checks

Finding ID: FYEO-BIO-02
Severity: **Informational**
Status: **Remediated**

Description

Some upper or lower bound checks are missing for user input values.

Proof of Issue

File name: programs/bio-launchpad/src/instructions/init_curation.rs
Line number: 81, 84

```
// Evaluate the starting time is now or in the the future
require!(timestamp_to_start >= current_timestamp,
BioError::CurationStartAtMustBeInTheFuture);
```

There is no upper bound on `timestamp_to_start` - it could be in a hundred years or more. This sets `curation_started_at` on the dao account.

```
// Evaluate the threshold is higher than the maximum contribution and the minimum is
lower than the maximum, if not, return an error
require!(threshold > maximum_contribution && minimum_contribution <=
maximum_contribution, BioError::CurationBadBoundaries);
```

There is no lower bound here. The `minimum_contribution` can be 0.

Severity and Impact Summary

All bounds should be considered to avoid mistakes. The `minimum_contribution` should be set so that the funding goal can actually be reached.

Recommendation

Make sure to add properly defined bounds for all values.

Tokens can only be withdrawn to ATA

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

Description

While this code is generally correct, for tokens not using Token2022 (token extensions), there may be associated token accounts (ATAs) that have their authority set to some other owner. This can happen due to the user falling for one of the most common scams on Solana.
The team has confirmed that the token does use Token2022 and therefore this concern does not apply.

Proof of Issue

File name: programs/bio-launchpad/src/instructions/withdrawal_curation.rs
Line number: 51

```
#[account(
    init_if_needed,
    payer = contributor,
    associated_token::mint = mint,
    associated_token::authority = contributor,
    associated_token::token_program = token_program,
)]
pub contributor_ata: InterfaceAccount<'info, TokenAccount>,
```

The `token::authority` could not be equal to `contributor`.

Severity and Impact Summary

This prohibits the user from withdrawing tokens should they have fallen victim to a scam. This would only apply to mints of the `Tokenkeg` program (the original token program).

Recommendation

If the mint is not using Token2022, consider supporting withdrawals for users that have been scammed and are no longer the authority of their ATA.

Transfer can be done with 0 amount

Finding ID: FYEO-BIO-04
Severity: **Informational**
Status: **Remediated**

Description

The withdraw function accepts 0 amounts.

Proof of Issue

File name: programs/bio-launchpad/src/instructions/withdrawal_curation.rs
Line number: 118

```
pub fn withdrawal(&mut self, amount: u64) -> Result<()> {
    ...


    // We call the CPI. Decimals are properties to the token account
    transfer_checked(ctx, amount, self.mint.decimals)?;
```

Severity and Impact Summary

This does nothing of use but burns gas.

Recommendation

Consider adding checks against 0 amounts.

Unfinished code

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

Description

The code is not complete, as indicated by several TODO comments in the codebase. These unfinished sections suggest missing logic related to bonding phase validation and account implementation.

Proof of Issue

File name: programs/bio-launchpad/src/instructions/withdrawal_curation.rs
Line number: 72

```
// TODO: This validation has to be updated when we will implement the bonding phase,
including the locking period of X days by parameter.
```

File name: programs/bio-launchpad/src/state/bonding.rs
Line number: 5

```
/// TODO: This account is not been developed yet. Currently it's a WIP
```

Severity and Impact Summary

These incomplete sections impact core functionality.

Recommendation

Complete the code.

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