FYEO

Security Code Review 1INTRO

1INTRO

April 2024
Version 1.0

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

Security Level
Public

Table of Contents

Executive Summary

Overview

1INTRO engaged FYEO Inc. to perform a Security Code Review of 1INTRO.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on March 21 - March 27, 2024, 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-1INTRO-01 – Admin can pass in an insecure auction token account in pool creation
- FYEO-1INTRO-02 – Any amount of fee accounts can be used, no delegate check
- FYEO-1INTRO-03 – Missing zero amount checks
- FYEO-1INTRO-04 – UpdateWeightsGradually could set weight to 0
- FYEO-1INTRO-05 – Insecure initialization of metadata state
- FYEO-1INTRO-06 – Token accounts are meant to be frozen but test relies on stale data
- FYEO-1INTRO-07 – Weight adjustment over time has no upper bound
- FYEO-1INTRO-08 – Wrong event emitted
- FYEO-1INTRO-09 – 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 1INTRO. 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/1intro/1intro-program with the commit hash 8f04afb527b40386cb1986d0e73a5ac304da02c2.

Commit hash of remediated code: ab145f8b08fe01f4dd5e8c6afeb246534d950aa5.

Files included in the code review

```
1intro-program/
└── programs/
    └── one-intro/
        └── src/
            ├── admins/
            │   ├── admin_update_admin_auth.rs
            │   ├── admin_update_fee_setting.rs
            │   ├── admin_withdraw_token.rs
            │   └── mod.rs
            ├── instructions/
            │   ├── create_metadata_state.rs
            │   ├── create_pool_state.rs
            │   ├── exit_pool.rs
            │   ├── join_pool.rs
            │   ├── mod.rs
            │   ├── poke_weights.rs
            │   ├── redeem_auction_token.rs
            │   ├── set_public_swap.rs
            │   ├── set_swap_fee.rs
            │   ├── swap_exact_amount_in.rs
            │   ├── swap_exact_amount_out.rs
            │   ├── update_token0_weight.rs
            │   ├── update_token1_weight.rs
            │   └── update_weights_gradually.rs
            ├── states/
            │   ├── metadata_state.rs
            │   ├── mod.rs
            │   └── pool_state.rs
            ├── utils/
```

Files included in the code review
```
│     ├── calc.rs
│     ├── consts.rs
│     ├── events.rs
│     └── mod.rs
├── error.rs
└── lib.rs
```

Table 1: Scope

Technical Analyses and Findings

During the Security Code Review 1INTRO, we discovered:

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

The following chart displays the findings by severity.

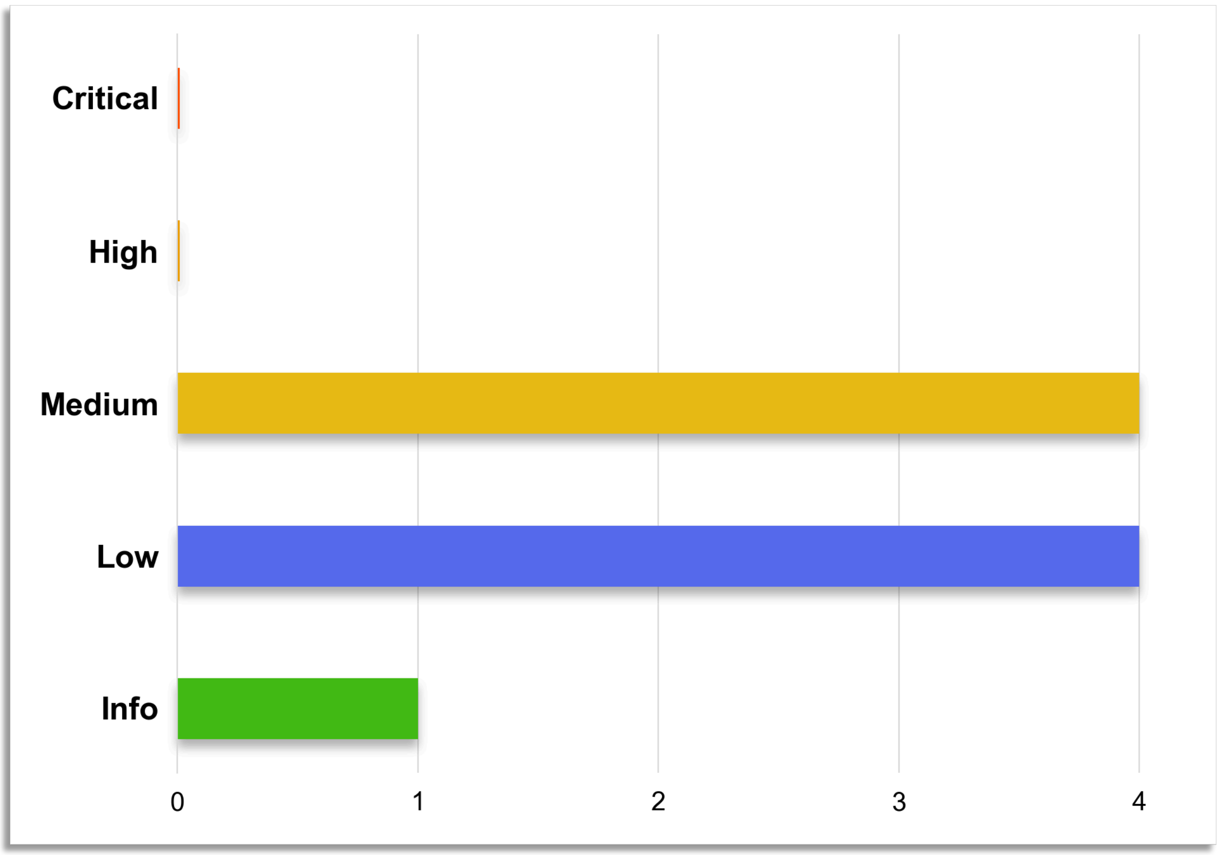

Figure 1: Findings by Severity

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
FYEO-1INTRO-01	**Medium**	Admin can pass in an insecure auction token account in pool creation
FYEO-1INTRO-02	**Medium**	Any amount of fee accounts can be used, no delegate check
FYEO-1INTRO-03	**Medium**	Missing zero amount checks
FYEO-1INTRO-04	**Medium**	UpdateWeightsGradually could set weight to 0
FYEO-1INTRO-05	**Low**	Insecure initialization of metadata state
FYEO-1INTRO-06	**Low**	Token accounts are meant to be frozen but test relies on stale data
FYEO-1INTRO-07	**Low**	Weight adjustment over time has no upper bound
FYEO-1INTRO-08	**Low**	Wrong event emitted
FYEO-1INTRO-09	**Informational**	Code clarity

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

Throughout the review process, it became evident that the codebase is meticulously organized, reflecting a commendable level of structural integrity. Notably, significant attention has been devoted to testing, with a comprehensive suite of tests in place, including negative tests aimed at verifying the resilience of the system against erroneous inputs and behaviors.

The adept utilization of the Anchor framework is evident throughout the code, particularly in the judicious application of its account constraint macros. These macros, conscientiously employed, serve as robust safeguards, bolstering the security posture of the program. Moreover, the incorporation of custom errors alongside account constraints not only enhances clarity but also demonstrates a thoughtful consideration for fellow developers, facilitating smoother comprehension and debugging processes.

An aspect deserving of acknowledgment is the responsiveness exhibited by the development team during the review period. Their prompt and attentive addressing of raised questions and concerns underscores a commitment to collaborative refinement and continuous improvement.

In summary, the codebase under review exemplifies a commendable synthesis of meticulous organization, rigorous testing, proficient utilization of framework capabilities, and a responsive collaborative ethos, all of which collectively contribute to its overall robustness and quality.

The implementation of the new referrer swap fee sharing feature has been carried out securely. This feature was implemented with the commit hash b7a143922b298e529e459742d86bf6b452495318.

Admin can pass in an insecure auction token account in pool creation

Finding ID: FYEO-1INTRO-01
Severity: **Medium**
Status: **Remediated**

Description

The creator of a pool could potentially submit a pool token account with a delegate or close authority set, which could allow for others to transfer tokens.

Proof of Issue

File name: programs/one-intro/src/lib.rs
Line number: 321

```
#[account(
    mut,
    constraint = *pool_auth_pda.key == pool_auction_token_account.owner
@ErrorCode::ConstraintInvalidTAOwner,
    constraint = pool_auction_token_mint.key() == pool_auction_token_account.mint
@ErrorCode::ConstraintInvalidTAMint,
    constraint = pool_auction_token_account.to_account_info().lamports() >=
Rent::get()?.minimum_balance(spl_token::state::Account::LEN)
@ErrorCode::ConstraintMinTokenAccountRent,
    owner = anchor_spl::token::ID @ErrorCode::OwnerTokenProgramID,
)]
pub pool_auction_token_account: Account<'info, TokenAccount>
```

Severity and Impact Summary

If an account with a delegate is passed in, the delegate could withdraw tokens from the pools account at any time.

Recommendation

Make sure this account does not have a delegate or close authority set.

Any amount of fee accounts can be used, no delegate check

Finding ID: FYEO-1INTRO-02
Severity: **Medium**
Status: **Remediated**

Description

While extra rent can be claimed from these accounts, it might add overhead and make this program unnecessarily complex.

Proof of Issue

File name: programs/one-intro/src/lib.rs
Line number: 513, 623

```
#[account(
    mut,
    constraint =
pool_state.load()?.valid_user_token_account(&metadata_state.token_auth_pda_key,
params.token_in_index, &metadata_swap_fee_account.to_account_info())
@ErrorCode::ConstraintInvalidTokenAccount,
    owner = anchor_spl::token::ID @ErrorCode::OwnerTokenProgramID
)]
pub metadata_swap_fee_account: Account<'info, TokenAccount>
```

This only checks the account owner, token authority and mint. The delegate and close authority are not checked.

Severity and Impact Summary

Any fee account with the correct token mint and authority is valid. This introduces complexity when collecting fees. It may also be possible that some token account has a delegate which could mean that some fees could potentially be stolen.

Recommendation

Use the associated token account to simplify collecting fees.

References

Anchor provides the `associated_token::authority` for this.
https://docs.rs/anchor-lang/latest/anchor_lang/derive.Accounts.html

Missing zero amount checks

Finding ID: FYEO-1INTRO-03
Severity: **Medium**
Status: **Remediated**

Description

Some functions do not properly check user input and may attempt to transfer 0 tokens.

Proof of Issue

File name: programs/one-intro/src/instructions/swap_exact_amount_in.rs
Line number: -

There are no checks for 0 amounts for `params.token_in_amount` and `min_token_out_amount`.

File name: programs/one-intro/src/instructions/swap_exact_amount_out.rs
Line number: -

There are no checks for 0 amounts for `token_out_amount` and `token_in_amount` which may be calculated 0.

File name: programs/one-intro/src/instructions/exit_pool.rs
Line number: -
There are no checks for 0 amounts for `pool_in_amount` and `min_out_amounts`.

Severity and Impact Summary

In some cases there may be panics and others may waste gas.

Recommendation

Make sure to check for 0 amounts consistently.

UpdateWeightsGradually could set weight to 0

Finding ID: FYEO-1INTRO-04
Severity: **Medium**
Status: **Remediated**

Description

This function does not check if the new weights are the same as the current weights. In case that they are equal, 0 will be set. This could happen if just one weight is meant to be updated over time.

Proof of Issue

File name: programs/one-intro/src/instructions/poke_weights.rs
Line number: 61

```
let mut new_weight = 0;

if start_weight > end_weight { ... }
else if start_weight < end_weight { ... }

pool_state.update_token_weight(index, new_weight)?;
```

Severity and Impact Summary

If a pool creator attempts to update just one weight by setting the other to it’s current value, they will inadvertently set the other to 0.

Recommendation

Make sure to not set the weight to 0 if the user does not intend to do so.

Insecure initialization of metadata state

Finding ID: FYEO-1INTRO-05
Severity: **Low**
Status: **Remediated**

Description

The initialization is done using a public function. There is a risk someone else can call it first.

Proof of Issue

File name: programs/one-intro/src/lib.rs
Line number: 140

```
#[account(
    mut,
    owner = system_program::ID @ErrorCode::OwnerSystemProgramID,
)]
/// CHECK: pass in admin auth account
pub admin: Signer<'info>,
```

Severity and Impact Summary

Anyone can initialize this contract.

Recommendation

For upgradeable contracts, it is recommended to use the following pattern to initialize the contract.

```
pub owner: Signer<'info>,
#[account(constraint = program.programdata_address()? == Some(program_data.key()))]
pub program: Program<'info, MyProgram>,
#[account(constraint = program_data.upgrade_authority_address == Some(owner.key()))]
pub program_data: Account<'info, ProgramData>,
```

Token accounts are meant to be frozen but test relies on stale data

Finding ID: FYEO-1INTRO-06
Severity: **Low**
Status: **Remediated**

Description

The token0 in this contract is meant to be frozen but several checks rely on stale account data. If the amount of tokens in an account is updated, it has to be reloaded.

Proof of Issue

File name: programs/one-intro/src/states/pool_state.rs
Line number: 242, 365

```
if self.is_user_repr_token_account(source_token_account) {
    self.simple_freeze_account(pool_state_pubkey, pda_authority, token_mint,
source_token_account, token_program)?;
}

...

self.simple_burn_token(token_mint, token_account, user_authority, token_program,
amount)?;

// freeze target token account
self.simple_freeze_account(pool_state_pubkey, pda_authority,
&token_mint.to_account_info(), token_account, token_program)?;
```

The burn and transfer functions do not reload the token account:

```
pub fn simple_burn_token<'info>(...) -> Result<()> {
    let result = burn(
        CpiContext::new(
            token_program.clone(),
            Burn {
                mint: token_mint.to_account_info(),
                from: token_account.to_account_info(),
                authority: authority.to_account_info().clone(),
            },
        ),
        amount,
    );

    token_mint.reload()?;

    result
}

pub fn simple_transfer_token<'info>(...) -> Result<()> {
    transfer(
        CpiContext::new(
            token_program.to_account_info(),
            Transfer {
                from: source_token_account.to_account_info(),
                to: target_token_account.to_account_info(),
                authority: authority.to_account_info(),
            },
        ),
        amount,
    )
}
```

Severity and Impact Summary

This functionality may leave the account in an unexpected state. Any account used in a CPI should be reloaded if updated data is required.

Recommendation

Consider always freezing the account. Otherwise make sure the account data is reloaded as required.

Weight adjustment over time has no upper bound

Finding ID: FYEO-1INTRO-07
Severity: Low
Status: Remediated

Description

There is no upper bound for the duration of weight adjustments.

Proof of Issue

File name: programs/one-intro/src/lib.rs
Line number: 1164

```
#[account(
    mut,
    constraint = pool_state.load()?.pool_status != PoolStatusReBalanced as u64
@ErrorCode::ConstraintInvalidPoolStatus,
    constraint = pool_state.load()?.pool_token_count == 2
@ErrorCode::ConstraintInvalidTokenCount,
    constraint = params.end_time.checked_sub(params.start_time).unwrap() >=
MIN_WEIGHT_CHANGE_TIME_PERIOD @ErrorCode::ConstraintInvalidWeightChangeSlotPeriod,
    constraint = params.end_time > Clock::get()?.unix_timestamp as u64
@ErrorCode::ConstraintInvalidEndTimestamp,
    constraint = params.target_weights[0] >= MIN_WEIGHT && params.target_weights[0] <=
MAX_WEIGHT @ErrorCode::ConstraintInvalidTokenWeight,
    constraint = params.target_weights[1] >= MIN_WEIGHT && params.target_weights[1] <=
MAX_WEIGHT @ErrorCode::ConstraintInvalidTokenWeight,
    owner = ID @ErrorCode::OwnerMyProgramID,
)]
pub pool_state: AccountLoader<'info, PoolState>,
```

Severity and Impact Summary

An upper limit may be worth considering to avoid user mistakes.

Recommendation

Consider adding an upper bound on the duration.

Wrong event emitted

Finding ID: FYEO-1INTRO-08
Severity: **Low**
Status: **Remediated**

Description

The code emits the wrong event `UpdateToken0WeightEvent` when the weight of Token1 is updated.

Proof of Issue

File name: programs/one-intro/src/instructions/update_token1_weight.rs
Line number: 72

```
emit!(UpdateToken0WeightEvent {
    event_name: "update_token1_weight".to_string(),
    new_weight: params.new_weight,
    delta_weight: delta_weight,
    delta_balance: delta_balance,
    delta_pool_amount: delta_pool_amount,
    token_balance: pool_state.pool_token_array[1].balance,
    token_weight: pool_state.pool_token_array[1].weight,
    pool_token_total_weight: pool_state.pool_token_total_weight,
});
```

Severity and Impact Summary

Programs that rely on this data may not work properly.

Recommendation

Use the `UpdateToken1WeightEvent` for Token1.

Code clarity

Finding ID: FYEO-1INTRO-09
Severity: **Informational**
Status: **Remediated**

Description

Some of the code does not follow best practices regarding rust code style and should be improved for better conciseness and improved maintainability.

Proof of Issue

File name: programs/one-intro/src/utils/calc.rs
Line number: 51,58, 65, 72, 79

```
pub fn add(left: f64, right: f64) -> f64 {
    ...
    return result;
}
```

At the end of the functions values can be returned by just writing `result`. Several instances of this pattern exist.

File name: programs/one-intro/src/instructions/update_token1_weight.rs
Line number: 75

```
delta_weight: delta_weight,
delta_balance: delta_balance,
delta_pool_amount: delta_pool_amount,
```

These can be assigned by just writing out the name once:

```
delta_weight,
delta_balance,
delta_pool_amount,
```

Several other places exist where this pattern can be simplified.

File name: programs/one-intro/src/admins/admin_withdraw_token.rs
Line number: 16

```
self.metadata_state.token_auth_pda_bump as u8
```

A u8 value does not need to be cast to u8.

File name: programs/one-intro/src/lib.rs
Line number: 399

```
constraint = creator_lp_token_account.to_account_info().lamports() >=
Rent::get()?.minimum_balance(spl_token::state::Account::LEN)
@ErrorCode::ConstraintMinTokenAccountRent,
```

This test occurs in many other places. Rent exemption checks can be enforced in a more concise manner by anchor:

```
rent_exempt = enforce
```

Token accounts are also rent exempt:
https://github.com/solana-labs/solana-program-library/blob/c21b4ae24aada1f766aae374f1ee9a42ae90cb98/token/program/src/processor.rs#L47
https://github.com/solana-labs/solana-program-library/blob/c21b4ae24aada1f766aae374f1ee9a42ae90cb98/token/program/src/processor.rs#L107

File name: programs/one-intro/src/utils/calc.rs
Line number: 169

```
let foo = sub(pow(y, weight_ratio), 1.0 as f64);
```

These can be written as `1.0` or just `1.`, no cast is required.

Severity and Impact Summary

Convoluted code decreases the maintainability of the codebase.

Recommendation

Simplify the code to help with maintainability.

Our Process

Methodology

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

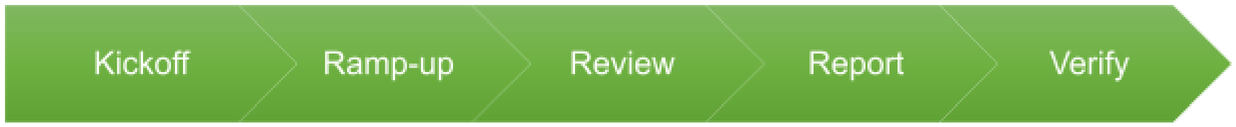

Figure 2: Methodology Flow

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