F Y E O

Security Code Review 4Cast Programs

4CAST

August 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

4CAST engaged FYEO Inc. to perform a Security Code Review 4Cast Programs.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on July 24 - July 30, 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-4CAST-01 – Admin can close any user state for rent
- FYEO-4CAST-02 – Incomplete bound checks
- FYEO-4CAST-03 – Instruction does not require signer
- FYEO-4CAST-04 – Insecure initialization
- FYEO-4CAST-05 – Code clarity
- FYEO-4CAST-06 – Duplicate account checks
- FYEO-4CAST-07 – Events emitted without action
- FYEO-4CAST-08 – Missing checks for 0 amounts

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 4Cast Programs. 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/4Cast-Studio/4Cast-Programs with the commit hash fe17284cc86d8e0705153197a9b7d858320d01a3.

Remediations were committed with the hash 88729ed38bc79e7f29252870b2fee3c1c08d9e51.

Files included in the code review

```
4cast-programs/
└── programs/
    ├── oracle-program/
    │   └── src/
    │       ├── defines/
    │       │   ├── consts.rs
    │       │   ├── errors.rs
    │       │   └── mod.rs
    │       ├── records/
    │       │   ├── global_state.rs
    │       │   ├── mod.rs
    │       │   ├── price_state.rs
    │       │   └── pyth_update_price_state.rs
    │       ├── transactions/
    │       │   ├── admin_modify_admin.rs
    │       │   ├── create_global_state.rs
    │       │   ├── create_pyth_price_state.rs
    │       │   ├── mod.rs
    │       │   └── update_pyth_price.rs
    │       └── lib.rs
    ├── prediction-program/
    │   └── src/
    │       ├── defines/
    │       │   ├── bpf_writer.rs
    │       │   ├── consts.rs
    │       │   ├── errors.rs
    │       │   └── mod.rs
    │       ├── records/
    │       │   ├── epoch_state.rs
    │       │   ├── global_state.rs
    │       │   ├── mod.rs
```

Files included in the code review

```
│   │   ├── user_state.rs
│   │   └── vault_state.rs
│   ├── transactions/
│   │   ├── admin_modify_admin.rs
│   │   ├── admin_modify_duration.rs
│   │   ├── admin_modify_fee_ratio.rs
│   │   ├── admin_modify_operation_latency.rs
│   │   ├── admin_modify_settlement_record.rs
│   │   ├── admin_withdraw_treasury.rs
│   │   ├── close_epoch_state.rs
│   │   ├── close_user_state.rs
│   │   ├── create_epoch_state.rs
│   │   ├── create_global_state.rs
│   │   ├── create_user_state.rs
│   │   ├── create_vault_state.rs
│   │   ├── deposit_fund.rs
│   │   ├── lock_epoch.rs
│   │   ├── mod.rs
│   │   ├── settle_epoch.rs
│   │   ├── settle_fund.rs
│   │   ├── settle_fund_v2.rs
│   │   ├── settle_fund_v3.rs
│   │   ├── update_community_wallet.rs
│   │   ├── update_development_wallet.rs
│   │   └── withdraw_fund.rs
│   └── lib.rs
└── reward-program/
    └── src/
        ├── defines/
        │   ├── errors.rs
        │   └── mod.rs
        ├── records/
        │   ├── mod.rs
        │   ├── reward_state.rs
        │   └── user_state.rs
        ├── transactions/
        │   ├── admin_modify_admin.rs
        │   ├── admin_modify_operator.rs
        │   ├── admin_modify_timelock_duration.rs
        │   ├── admin_pause_user.rs
        │   ├── close_user_state.rs
        │   ├── create_reward_state.rs
        │   ├── create_user_state.rs
```

Files included in the code review
│ ├── mod.rs │ ├── upload_reward.rs │ └── withdraw_reward.rs └── lib.rs

Table 1: Scope

Technical Analyses and Findings

During the Security Code Review 4Cast Programs, we discovered:

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

The following chart displays the findings by severity.

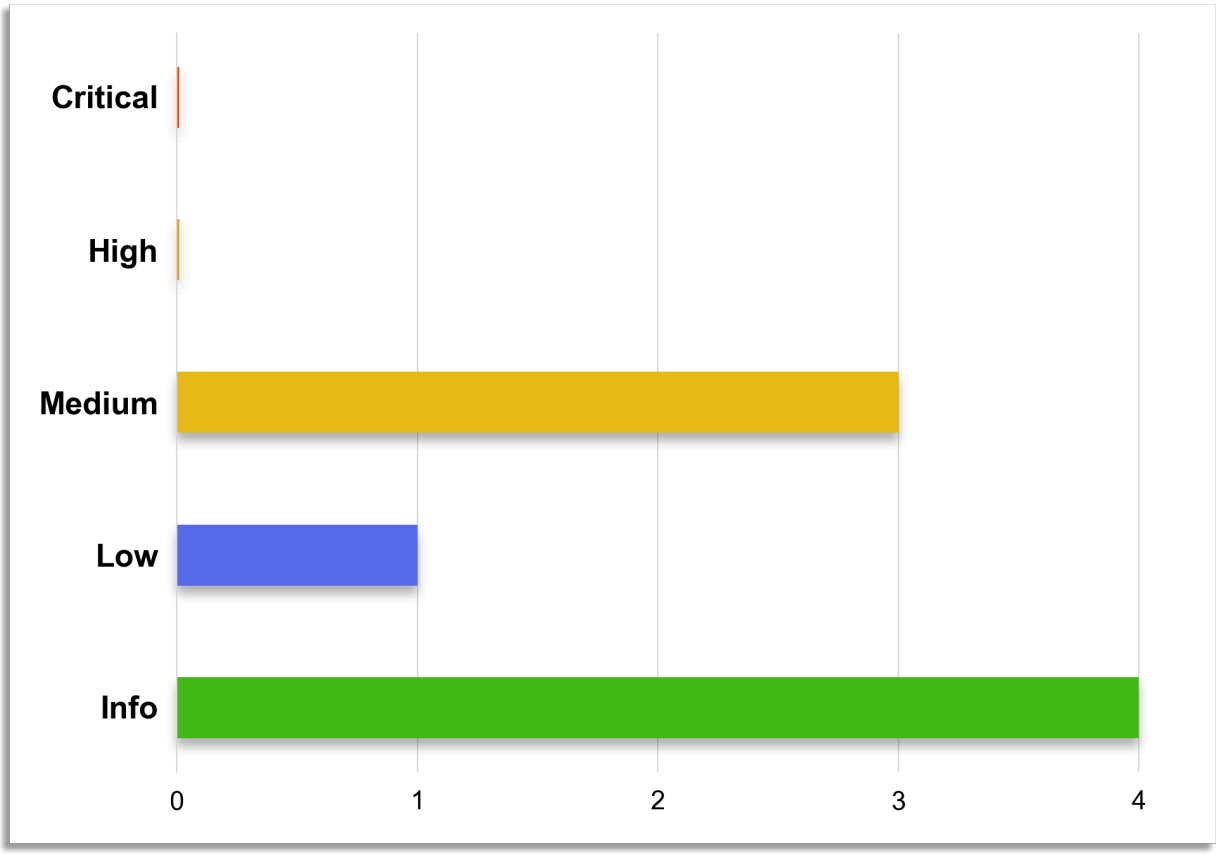

Figure 1: Findings by Severity

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-4CAST-01	**Medium**	Admin can close any user state for rent
FYEO-4CAST-02	**Medium**	Incomplete bound checks
FYEO-4CAST-03	**Medium**	Instruction does not require signer
FYEO-4CAST-04	**Low**	Insecure initialization
FYEO-4CAST-05	**Informational**	Code clarity
FYEO-4CAST-06	**Informational**	Duplicate account checks
FYEO-4CAST-07	**Informational**	Events emitted without action
FYEO-4CAST-08	**Informational**	Missing checks for 0 amounts

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 4cast Solana program is a decentralized prediction market platform composed of three main components: an oracle adapter, a rewards program, and a prediction program. The oracle adapter integrates Pyth price feeds to provide accurate and timely asset price data. The prediction program manages the core functionality of creating and resolving prediction markets, allowing users to forecast token price movements within specific time periods. The rewards program handles the distribution of incentives to participants, including market creators, operators, and settlers.

The platform leverages Solana's high-performance blockchain to ensure all operations are fully on-chain, promoting transparency and fairness. It implements a self-custody model, allowing users to participate directly from their wallets. The prediction mechanism is structured around timed rounds, with users predicting whether prices will go up or down.

4cast incorporates a permissionless market creation feature, enabling users to establish new prediction markets for various crypto assets. The program utilizes smart contracts to automate market operations, including entry phases, price locking, and outcome determination. Market operators and settlers play crucial roles in processing predictions and resolving outcomes, interacting with the program to maintain market functionality.

Overall, 4cast aims to provide a robust, decentralized infrastructure for crypto prediction markets on the Solana blockchain.

The review of this Solana program revealed that the code is well-developed and organized. It effectively uses the Anchor framework and maintains a clear structure. The team has been responsive and communicated well during the review process, which has helped make the review efficient and collaborative.

Admin can close any user state for rent

Finding ID: FYEO-4CAST-01
Severity: **Medium**
Status: **Remediated**

Description

The `CloseUserState` instruction does not check if the `user_state` belongs to the `reward_state` that the admin is an admin for.

Proof of Issue

File name: programs/reward-program/src/transactions/close_user_state.rs
Line number: 24

Severity and Impact Summary

This makes it possible that any admin can immediately close `user_state` and collect rent refunds for doing so. This could be used to do a DoS attack.

Recommendation

Make sure that only the admin for this particular `reward_state` can close accounts.

Incomplete bound checks

Finding ID: FYEO-4CAST-02
Severity: **Medium**
Status: **Remediated**

Description

Some parameters are not properly checked for bounds. Some of these parameters could cause issues when misconfigured by accident or with malicious intent.

Proof of Issue

File name: programs/prediction-program/src/records/epoch_state.rs
Line number: 182

```
if current_time >=
self.lock.time.checked_add(vault.vault_duration).unwrap().checked_add(global.vault_max_o
peration_latency).unwrap() {
    return EpochStatusUnaccomplished as u64;
}

pub fn valid_operation_latency<'info>(&self, operation_latency: u64) -> bool {
    operation_latency <= MAX_OPERATION_LATENCY
}
```

This could be set to 0 which would effectively cause a DoS situation as no epochs could successfully conclude. This can only be done by the admin however.

File name: programs/prediction-program/src/records/global_state.rs
Line number: 65

```
pub fn valid_duration<'info>(&self, duration_index: u8, duration: u64) -> bool {
    self.vault_duration_list[duration_index as usize] == duration
}

#[account(
    seeds = [GlobalState::GLOBAL_STATE_PDA_SEED],
    bump,
    constraint = global_state.valid_duration(parameters.duration_index,
parameters.duration) @ErrorInvalidDuration,
)]
pub global_state: Box<Account<'info, GlobalState>>,

self.global_state.vault_duration_list[0] = 60;
self.global_state.vault_duration_list[1] = 120;
self.global_state.vault_duration_list[2] = 180;
self.global_state.vault_duration_list[3] = 300;
self.global_state.vault_duration_list[4] = 600;
self.global_state.vault_duration_list[5] = 0;
self.global_state.vault_duration_list[6] = 0;
self.global_state.vault_duration_list[7] = 0;
```

This allows for the duration to be 0.

File name: programs/reward-program/src/records/reward_state.rs
Line number: 31

```
pub fn is_valid_timelock_duration(&self, timelock_duration: u64) -> bool {
    timelock_duration <= 86400 * 30
}
```

No lower bound check

Severity and Impact Summary

Some misconfigurations could cause DoS situations which may lead to unhappy users.

Recommendation

Make sure to verify that all parameters are within reasonable bounds.

Instruction does not require signer

Finding ID: FYEO-4CAST-03
Severity: **Medium**
Status: **Remediated**

Description

This instruction is not signed by anyone.

Proof of Issue

File name: programs/prediction-program/src/transactions/close_user_state.rs
Line number: 16

Severity and Impact Summary

This function can be executed on behalf of any user. This could enable a DoS attack on individual users preventing them from using the app.

Recommendation

Make sure to check the user has signed this instruction.

Insecure initialization

Finding ID: FYEO-4CAST-04
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/prediction-program/src/transactions/create_global_state.rs
Line number: 86

```
pub struct CreateGlobalState<'info> {
    ...

    #[account(mut)]
    /// CHECK: admin signer
    pub admin: Signer<'info>,
```

File name: programs/oracle-program/src/transactions/create_global_state.rs
Line number: 86

```
pub struct CreateGlobalState<'info> {
    ...

    #[account(mut)]
    /// CHECK: admin signer
    pub admin: Signer<'info>,
```

Severity and Impact Summary

Anyone can initialize these programs.

Recommendation

For upgradeable programs, it is recommended to use the following pattern to initialize the program.

```
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>,
```

Code clarity

Finding ID: FYEO-4CAST-05
Severity: **Informational**
Status: **Remediated**

Description

In some instances the code could be improved for readability.

Proof of Issue

File name: programs/prediction-program/src/transactions/admin_withdraw_treasury.rs
Line number: 37, 52

```
constraint = *treasury_auth_pda.key == treasury_token_account.owner @ErrorInvalidTAOwner

...

constraint = *admin.key == admin_token_account.owner @ErrorInvalidTAOwner,
```

Use the `token::authority` check provided by Anchor.

File name: programs/oracle-program/src/records/price_state.rs
Line number: 22

```
pub enum PriceSource { ... }

pub enum PriceType { ... }

#[account]
#[derive(Debug, Default)]
pub struct PriceState {
    pub price_source: u8,
    pub price_type: u8,
    ...
}
```

It looks like the enums are declared here but not actually used in the `PriceState` struct.

File name: programs/oracle-program/src/records/global_state.rs
Line number: 10

```
pub struct GlobalState {
    pub admin_key: Pubkey,
    pub global_state_pda_bump: u8,

    // 32 more u64 reserved
    pub _reserved: [u64; 32],
}
```

Using arrays of reserved bytes has caused high CPU usage for Anchor front-ends when deserializing accounts. The account size is given manually: `space = 8 + std::mem::size_of::<GlobalState>()`, it is recommended to do extra padding here rather than adding dummy data. The final size could also be declared as a constant in `impl GlobalState`.

Severity and Impact Summary

Not a security concern.

Recommendation

Keeping the code concise can help with readability and maintenance.

Duplicate account checks

Finding ID: FYEO-4CAST-06
Severity: **Informational**
Status: **Remediated**

Description

Some of the account checks are duplicated. For example, both the `Token` program and `System` program wrappers check the account key. Additional checks using macros are therefore duplicates.

Proof of Issue

File name: across many files ~12 occurrences

```
#[account(
    address = system_program::ID
)]
pub system_program: Program<'info, System>,

#[account(
    address = anchor_spl::token::ID
)]
pub token_program: Program<'info, Token>,
```

Severity and Impact Summary

No security impact.

Recommendation

The readability of the code could be slightly improved by removing unnecessary checks.

Events emitted without action

Finding ID: FYEO-4CAST-07
Severity: **Informational**
Status: **Remediated**

Description

Some events will be emitted regardless if anything of interest was done or not.

Proof of Issue

File name: programs/oracle-program/src/transactions/update_pyth_price.rs
Line number: 55

```
let current_slot = Clock::get()?.slot;
if current_slot > self.price_state.last_updated_slot {
    self.price_state.last_updated_slot = current_slot;
    (self.price_state.price_value, self.price_state.last_updated_timestamp) =
self.pyth_update_price_state.get_price(&self.pyth_main_price_feed_account,
&self.pyth_base_price_feed_account);
}

emit!(UpdatePythPriceEvent { ... }
```

File name: programs/prediction-program/src/transactions/withdraw_fund.rs
Line number: 97

```
let withdraw_amount = std::cmp::min(parameters.amount,
self.user_state.user_token_amount);
if withdraw_amount > 0 {
    self.user_state.user_token_amount =
self.user_state.user_token_amount.checked_sub(withdraw_amount).unwrap();
    ...
}

emit!(WithdrawFundEvent {
```

Severity and Impact Summary

No security impact. It may however confuse 3rd party developers processing these events.

Recommendation

Consider creating events only when something of interest has happened.

Missing checks for 0 amounts

Finding ID: FYEO-4CAST-08
Severity: **Informational**
Status: **Remediated**

Description

There are some instances in which amounts should be checked against 0.

Proof of Issue

File name: programs/prediction-program/src/records/user_state.rs
Line number: 69

```
pub fn can_deposit<'info>(&self, epoch: u64, parameters: &DepositFundParameters,
global_state: &Box<Account<'info, GlobalState>>) -> bool {
    let settlement_token_record =
&global_state.settlement_token_list[parameters.token_mint_index as usize];

    // deposit is not allowed if amount is less than min deposit amount
    if settlement_token_record.min_per_wallet_epoch_token_amount > 0 {
        if parameters.amount < settlement_token_record.min_per_wallet_epoch_token_amount
{
            return false;
        }
    }
    ...
```

While the current config has a `min_per_wallet_epoch_token_amount`, this may not be configured for all tokens.

File name: programs/reward-program/src/records/user_state.rs
Line number: 30

```
pub fn valid_upload_amount<'info>(&self, amount: u64, reward_token_account:
&Box<Account<'info, TokenAccount>>) -> bool {
    amount <= reward_token_account.amount
}
```

Used in `UploadReward`, this will do nothing of use when called with 0.

File name: programs/reward-program/src/transactions/withdraw_reward.rs
Line number: 74

```
impl<'info> WithdrawReward<'info> {
    pub fn process(&mut self) -> Result<()> {
        // calculate user total reward token amount
        let amount =
self.user_state.user_reward_token_amount.checked_add(self.user_state.user_last_updated_r
eward_token_amount).unwrap();
        self.user_state.user_reward_token_amount = 0;
        self.user_state.user_last_updated_reward_token_amount = 0;

        transfer( ... )
```

The amount may turn out to be `0`. There is no need to call the transfer instruction and deposit events.

Severity and Impact Summary

Unnecessary calls may be done, events may be created without need.

Recommendation

Make sure to check for 0 amounts.

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