F Y E O

Security Code Review Six Sigma

Saage

September 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

Saage engaged FYEO Inc. to perform a Security Code Review Six Sigma.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on August 22 - August 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-Saage-01 – Liquidity provided can be 1 token
- FYEO-Saage-02 – MarketId is not checked to be unique
- FYEO-Saage-03 – Participation cap can be set to 0
- FYEO-Saage-04 – All admins can be removed
- FYEO-Saage-05 – Potential division by 0
- FYEO-Saage-06 – Insufficient bound checks

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 Six Sigma. 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/saage-tech/six-sigma-v2 with the commit hash fe68e57d73353a9839f23fcabccb9e862215e1f4.

Remediations were provided with commit hash 18249812fa45b3b53a10a2b3541711291d7d36f3.

Files included in the code review

```
six-sigma-v2/
└── packages/
    ├── contract/
    │   └── src/
    │       ├── orderbook/
    │       │   ├── state/
    │       │   │   ├── account.rs
    │       │   │   ├── admin.rs
    │       │   │   ├── bet.rs
    │       │   │   ├── market.rs
    │       │   │   ├── mod.rs
    │       │   │   ├── monotonic_id.rs
    │       │   │   ├── prefix_sum.rs
    │       │   │   ├── rewards.rs
    │       │   │   └── sparse_queue.rs
    │       │   ├── fnv.rs
    │       │   ├── mod.rs
    │       │   ├── range.rs
    │       │   └── token.rs
    │       ├── strategic_reserve/
    │       │   ├── admin.rs
    │       │   └── mod.rs
    │       └── lib.rs
    ├── deposit-bonus-token/src/lib.rs
    ├── kyc-token/src/lib.rs
    └── signed-odds/src/lib.rs
```

Table 1: Scope

Technical Analyses and Findings

During the Security Code Review Six Sigma, we discovered:

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

The following chart displays the findings by severity.

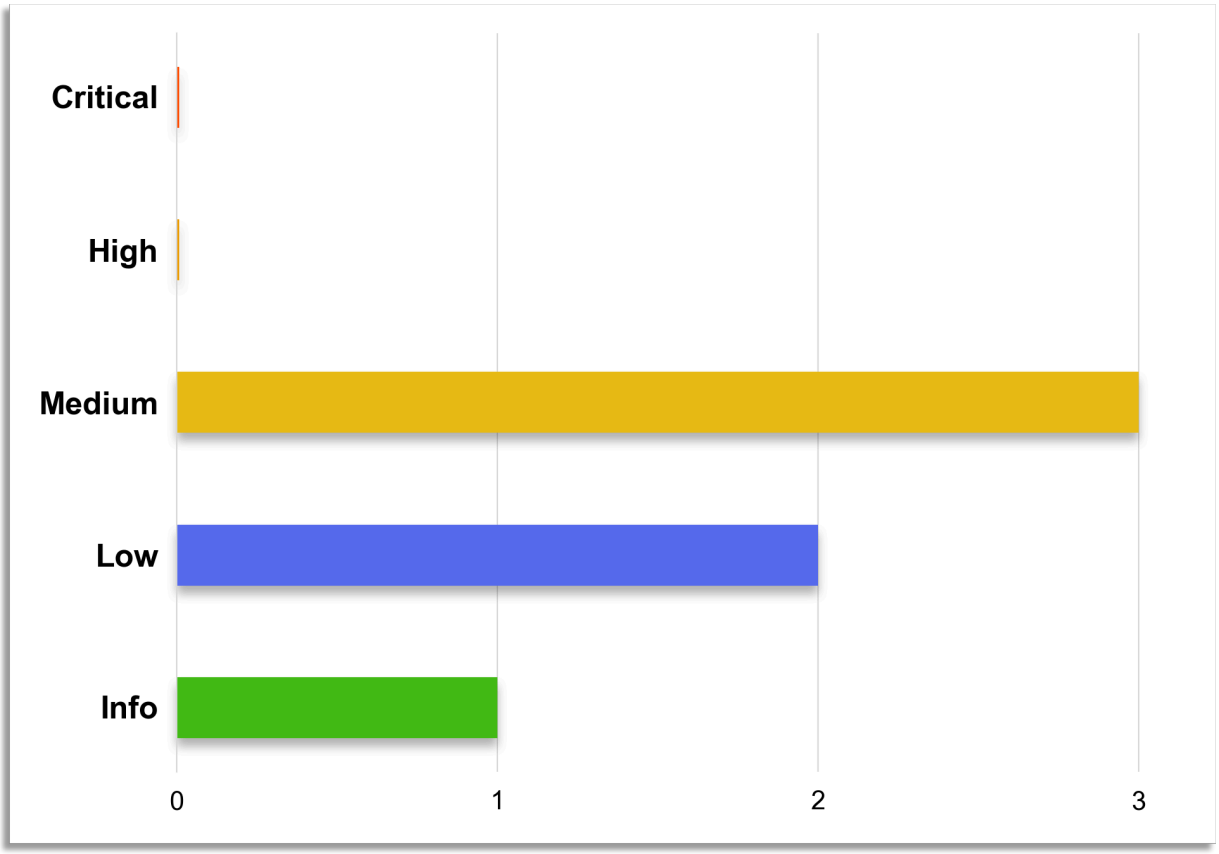

Figure 1: Findings by Severity

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-Saage-01	Medium	Liquidity provided can be 1 token
FYEO-Saage-02	Medium	MarketId is not checked to be unique
FYEO-Saage-03	Medium	Participation cap can be set to 0
FYEO-Saage-04	Low	All admins can be removed
FYEO-Saage-05	Low	Potential division by 0
FYEO-Saage-06	Informational	Insufficient bound checks

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

Six Sigma Sports is a sports betting platform built on Cosmos. The platform aims to democratize sports betting by leveraging blockchain technology to provide a transparent, fast, and uniquely engaging experience. The smart contract handles wager processing and automatic bet settlement, manages key operations such as market creation, user participation, and fund distribution. With its blockchain foundation, Six Sigma Sports ensures enhanced transparency and efficiency.

The platform consists of two main contracts: the strategic reserve and the betting platform. Administrators can create new betting markets, which draw funds from the strategic reserve. These markets can be in various states, including active, suspended, or settled, and can be designated as main markets or restricted by administrators.

To encourage platform usage, administrators manage various incentive campaigns. These campaigns offer bonuses for actions such as placing bets, referring new users, signing up, or making deposits. Participation in these campaigns requires authorization through an externally-provided signed token. User verification (KYC) is handled by an external service. Users must obtain signed tokens to interact with the contract, such as submitting bets.

Bets are queued for processing and include a slippage factor to account for potential odds changes. The odds for bets are provided separately and must be signed by an appropriate authority before the contract can process and integrate a pending bet into a market.
Authorized users can also act as a “house” by providing liquidity to markets of their choice, subject to certain fees.

The platform includes responsible gambling features, allowing users to set daily, weekly, and monthly betting limits on their accounts. A delay could be added to prevent immediate changes to these limits, further promoting responsible gambling.
Administrators play a crucial role in managing these contracts, with the ability to adjust various settings to ensure smooth operation of the platform.

The platform’s smart contracts are written in high-quality Rust code, demonstrating strong technical proficiency, although they could benefit from additional in-code documentation to further enhance clarity and maintainability.

Liquidity provided can be 1 token

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

Description

There is a limit on how many users can provide liquidity. But there is no lower bound on the amount. Thus 1 token or any other negligible amount of tokens can be provided.

Proof of Issue

File name: packages/contract/src/orderbook/mod.rs
Line number: 731

```
ExecuteMsg::HouseProvideLiquidity { token, market_id } => {
    let config = CONFIG.load(ctx.storage)?;
    let amount = assert_sent_amount(&info, &config)?,
    ...

...

if queue.len() == Self::MAX_HOUSE_USERS {
    return Err(Error::MarketFull {
        max: Self::MAX_HOUSE_USERS,
    });
}
```

Severity and Impact Summary

If the queue is filled with users providing 1 token each or any other minimal amount of liquidity, this will not be helpful to the system.

Recommendation

Make sure to check for some minimum amount provided.

MarketId is not checked to be unique

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

Description

The `Market::new()` function does not verify that the market id is not yet in use. If it is already in use, data will be overwritten.

Proof of Issue

File name: packages/contract/src/orderbook/state/market.rs
Line number: 350

```
ExecuteMsg::AddMarket {
    key,

...

let payload = AddMarketPayload {
    amount: amount.into(),
    key,

if Self::MARKET_DATA.has(ctx.storage, key) {
    return Err(Error::MarketAlreadyExists);

...

Self::MARKET_DATA.save(ctx.storage, market.key, &market.data)?;
Self::MARKETID_TO_KEY.save(ctx.storage, key.market_id, &key.prefix())?;
```

Severity and Impact Summary

This will essentially hide existing market data, as the key for that is based on the `MARKETID_TO_KEY` where the same id would then only point to one item and the other remains “hidden”.

Recommendation

Make sure to not overwrite existing market ids.

Participation cap can be set to 0

Finding ID: FYEO-Saage-03
Severity: Medium
Status: Remediated

Description

The participation cap can be set to 0 since `change_parameters()` does not check the input value but rather the currently set value before applying the input.

Proof of Issue

File name: packages/contract/src/orderbook/state/rewards.rs
Line number: 352

```
if let Some(cap) = params.cap {
    if self.data.cap == 0 {
        return Err(StdError::generic_err("Participation cap cannot be 0."));
    }

    self.data.cap = cap;
}
```

If `params.cap` is set, it will be applied unless `self.data.cap == 0`.

The `new()` function does check:

```
// TODO: Should probably enforce a maximum here.
if params.cap == 0 {
    return Err(StdError::generic_err("Participation cap cannot be 0."));
}
```

Severity and Impact Summary

Recommendation

Add a bounds test `Some(cap)` to establish it is greater than 0 and also find a suitable upper bound.

All admins can be removed

Finding ID: FYEO-Saage-04
Severity: Low
Status: Remediated

Description

It is possible to remove the last admin in the contracts. This would leave the contract unmanageable.

Proof of Issue

File name: packages/contract/src/orderbook/mod.rs
Line number: 1043

```
ExecuteMsg::RemoveAdmin { address } => {
    admin::assert_has_permission(ctx.storage, &info, Permissions::SETTINGS)?;

    let address = deps.api.addr_validate(&address)?;
    admin::remove(ctx.storage, &address);

    ctx.into_response()
}
```

File name: packages/contract/src/strategic_reserve/mod.rs
Line number: 172

```
ExecuteMsg::RemoveAdmin { address } => {
    admin::assert_has_permission(deps.storage, &info, Permissions::ADMIN)?;

    let address = deps.api.addr_validate(&address)?;
    admin::remove(deps.storage, &address);

    Response::new()
}
```

Severity and Impact Summary

By accident or with malicious intent, all admins could be removed.

Recommendation

Make sure the contracts can not end up in unmanageable states.

Potential division by 0

Finding ID: FYEO-Saage-05
Severity: **Low**
Status: **Remediated**

Description

The token decimal calculation does not work for more than 18 digits, as it will attempt to divide by 0. The configuration can be made with any number of digits however.

Proof of Issue

File name: packages/contract/src/orderbook/token.rs
Line number: 16

```
pub fn amount_raw(&self, value: Decimal256) -> StdResult<u128> {
    let result = if value.is_zero() {
        Uint128::zero()
    } else if self.decimals != 18 {
        let factor =
            Decimal256::one().atomics() /
Uint256::from_u128(10).pow(self.decimals.into());
        let raw = value.atomics() / factor;

        Uint128::try_from(raw)?
    } else {
        value.atomics().try_into()?
    };

    Ok(result.u128())
}
```

Severity and Impact Summary

This calculation would always run into errors if the contract is misconfigured.

Recommendation

Make sure to check for this case or implement the required functionality.

Insufficient bound checks

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

Description

The bounds of certain values are not established or are somewhat wide.

Proof of Issue

File name: packages/contract/src/orderbook/mod.rs
Line number: 480

```
if config.house_fee > Decimal256::one() {
    return Err(StdError::generic_err("House fee cannot be higher than 1.").into());
}

if config.bet_fee > Decimal256::one() {
    return Err(StdError::generic_err("Bet fee cannot be higher than 1.").into());
}
```

File name: packages/contract/src/orderbook/mod.rs
Line number: 951

```
if let Some(fee) = house_fee {
    if fee > Decimal256::one() {
        return Err(StdError::generic_err("House fee cannot be higher than 1.").into());
    }
    config.house_fee = fee;
}

if let Some(fee) = bet_fee {
    if fee > Decimal256::one() {
        return Err(StdError::generic_err("Bet fee cannot be higher than 1.").into());
    }
    config.bet_fee = fee;
}
```

Here the fees can add up to 200%.

File name: packages/contract/src/orderbook/state/rewards.rs
Line number: 254

```
// TODO: Should probably enforce a maximum here.
if params.cap == 0 {
    return Err(StdError::generic_err("Participation cap cannot be 0."));
}
```

As pointed out by the `TODO`, an upper bound is missing.

File name: packages/contract/src/orderbook/state/rewards.rs
Line number: 474

```
fn validate_params(&self) -> StdResult<()> {
    match self {
        Self::BetBonus {
            payout_percentage,
            bet_size,
            ..
        } => {
            if payout_percentage.is_zero() {
                ...
            }

            if bet_size.is_invalid() {
                ...
            }
        }
        Self::MainMarket {
            payout_percentage,
            bet_size,
        } => {
            if payout_percentage.is_zero() {
                ...
            }

            if bet_size.is_invalid() {
                ...
            }
        }
        Self::SignupAndDepositBonus { bonus_percentage } => {
            if bonus_percentage.is_zero() {
                ...
            }
```

Neither `payout_percentage` nor `bonus_percentage` have an upper bounds check. The `bet_size` only checks that the minimum does not exceed the maximum.

File name: packages/contract/src/orderbook/state/rewards.rs
Line number: 366

```
match params.unlock_criteria.duration {
    ParameterStateChange::Unchanged => {}
    ParameterStateChange::Remove => self.data.unlock_criteria.duration = None,
    ParameterStateChange::Set(duration) => {
        self.data.unlock_criteria.duration = Some(duration)
    }
}
```

No bounds are established.

Severity and Impact Summary

Choosing concise bounds can prevent mistakes or malicious changes.

Recommendation

Consider the requirements and choose bounds that best suit the expected use.

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