Pregame Solana Program Code Review and Security Analysis

Client: Pregame

Date: 11th April 2025

Version: 1.0

Table of Contents

Security Review Information

Repository	pregame
Initial commit	fca3dc77bcad9367d183ce8c080d2c6f6bd7da87
Scope	programs/pregame/*
Version	Final Report (v1.0)
Date	11th April 2025

Executive summary

As of April 11, 2025, our comprehensive security review of the Pregame protocol has been concluded. Initially, the assessment identified **4** critical-severity, **3** high-severity, **2** medium-severity, and **1** low-severity vulnerabilities. All the issues were fixed in the remediation part of the audit process.

Introduction

Torii Security has been commissioned by Pregame to conduct a comprehensive security review of their Solana Program, focusing on the robustness, security, and efficiency of the program's implementation. The review aims to critically evaluate the program's architecture and codebase, with the following specific objectives:

- **Verify Protocol Integrity**: Assess the program's operation against its design specifications to ensure it functions correctly and efficiently within the Solana ecosystem. This includes evaluating its interaction with other protocols and services on the Solana network.
- **Identify Security Vulnerabilities**: Uncover potential security weaknesses that could be exploited by malicious actors, including but not limited to, flaws in program logic, transaction handling, and external dependencies.
- **Detect Program Bugs**: Identify bugs and glitches in the code that may result in unintended or erratic program behavior, potentially compromising its performance or security.
- **Provide Improvement Recommendations**: Offer actionable advice to enhance the program's security posture, efficiency, and code clarity, aiming to fortify it against current and future security threats while improving maintainability and scalability.

Exclusions from This Review

While the security review conducted by Torii Security on behalf of Pregmae provides a comprehensive analysis of the Solana Program's architecture, codebase, and security posture, certain aspects are beyond the scope of this audit. These exclusions are critical for stakeholders to understand, as they may require separate consideration and evaluation. The following areas have not been verified as part of this security review:

- **Deployment and Program Upgrade Process:** The procedures and mechanisms for deploying the Solana Program to the network and subsequent upgrades or modifications to the program are not covered. This includes the validation of deployment scripts, migration strategies, and the security of upgradeable contract mechanisms.
- **Keys Management**: The management, storage, and security practices for keys, including administrator keys and those used for program interactions, are outside the scope of this review. This encompasses both the technical and procedural safeguards in place to protect keys from unauthorized access or misuse.
- **Economic Vulnerabilities**: The review does not delve into the economic aspects or incentive structures of the Pregame protocol. Potential vulnerabilities arising from economic models, tokenomics, or financial incentives that could impact the program's security or integrity are not evaluated.

Protocol Overview

The Pregame P2P Betting Protocol is a decentralized betting platform that allows users to create and oppose wagers on sporting events. Participants lock tokens in dedicated escrow accounts, from which winners receive payouts once the bet is settled. The protocol's design aims to ensure secure custody of user funds during the betting lifecycle and employs administrative checks and settlement logic to prevent abuses or misconfigurations. Administrative privileges are vested in an admin role, which manages protocol-wide configurations, bet settlement, treasury updates, and token whitelisting.

Key Features

- **Decentralized Escrow Mechanism –** Users deposit tokens (from whitelisted mints) into program-owned escrow accounts when creating or joining (opposing) a bet. This design isolates user funds from personal wallets until the bet is settled**.**
- **Lockup Presets (Bet Creation Parameters)** – New bets follow specific parameters such as minimum wager, odds, and total opposing capacity. These constraints help maintain consistent user experiences and limit misconfiguration.
- **Settlement Operations** – After an event concludes, an authorized party (the admin) finalizes the bet, distributing escrowed funds to winners and taking a small fee for the protocol.
- **Odds and Rewards** – The protocol calculates potential winnings based on provided odds. When bets are settled, winners receive their original stake plus a share of the total escrow, less protocol fees.
- **Dynamic Deposits and Withdrawals** – Opponents can enter a bet if it remains open and has remaining "action capacity." Once it is fully opposed or the event commences, no new positions can be taken.

Risks

Below are the risks identified based on the operational and procedural aspects of the Pregame Betting Protocol. While the underlying code has been reviewed for major security pitfalls, these points highlight areas that may lie beyond strict code-level vulnerabilities.

- **Superadmin Dependency:** The admin role can update treasuries, settle bets, and whitelist token mints. If compromised, an attacker could forcibly settle bets, confiscate funds, or disrupt the protocol.
- **Settlement Coordination:** Settlement relies on off-chain calls to confirm match outcomes and finalize bets. Errors or delays in calling these settlement instructions can result in locked user funds or incorrect distribution of winnings.
- **Processing Delays (Withdrawal/Settlement Requests):** Settlement may require manual admin intervention. Users could face payout delays if the admin is unavailable or unresponsive, damaging user experience and trust.
- **Frontend Risks:** As with any decentralized protocol, the frontend interface plays a critical role in user interactions. Any compromise in the frontend could expose users to phishing attacks, unauthorized transactions, or incorrect operations.

- **Deployment and Upgradability Risks:** As with any decentralized protocol, errors or vulnerabilities during the deployment phase or in upgradeability features can introduce critical security flaws. An improper upgrade or deployment could expose the system to attacks or lead to incorrect functioning of key operations.

Methodology

Issue Severity Classification

This report differentiates identified issues into distinct severity levels, each reflecting the potential impact on the system's security and overall functionality.

Severity	**Description**
Critical	Issues that present an immediate and severe threat, such as significant financial loss, irreversible locking of funds, or catastrophic system failure. These vulnerabilities require urgent remediation.
High	Bugs or vulnerabilities that could disrupt the correct operation of the system, potentially leading to incorrect states or temporary denial of service. Prompt attention and corrective action are necessary.
Medium	Issues that indicate deviations from best practices or suboptimal use of system primitives. While they may not pose immediate security threats, these issues could lead to vulnerabilities or inefficiencies if unaddressed.
Low	Minor concerns that have anegligible impact on system security or functionality. These may include inefficiencies or minor deviations from best practices that are unlikely to affect the system's operation significantly.
Informational	Suggestions related to design decisions, potential enhancements, or optimizations that do not have adirect impact on security. Implementing these recommendations may improve aspects such as usability or code readability but is not essential for system security.

Issue Status Definitions

Each issue is assigned a status reflecting its current resolution stage.

Status	**Description**
Pending	The issue has been identified but not yet reviewed or addressed by the development team.
Acknowledged	The development team has recognized the issue but has not completed its resolution.
Resolved	The issue has been fully addressed, with implemented changes verified for effectiveness.

Findings Summary

ID	Title	Severity	Status
TS-C1	Opponents Can Settle and Claim Rewards Even If They Lost	**Critical**	**Resolved**
TS-C2	Missing Mint Validation in Settlement Allows Griefing and Creator Exploit to Steal Escrowed Tokens	**Critical**	**Resolved**
TS-C3	Missing Mint Consistency Check Allows Bets with Mismatched or Fake Tokens	**Critical**	**Resolved**
TS-C4	No Access Control on SettleBet Allows Anyone to Arbitrarily Pick a Winner	**Critical**	**Resolved**
TS-H1	DoS Risk: init Macro Fails if Malicious User Pre-initializes PDA Token Account	**High**	**Resolved**
TS-H2	Lack of Token-2022 Extension Validation Can Lead to Transfer Hook Abuse or Fee Miscalculations	**High**	**Resolved**
TS-H3	Missing Treasury Wallet Validation Allows Fee Redirection by Arbitrary Users	**High**	**Resolved**
TS-M1	Insecure update_bet Instruction Allows Arbitrary Opponent Count Modification	**Medium**	**Resolved**
TS-M2	Using ATA Authority in Constraints Allows Bet Creator to Block Settlements by Changing Account Authority	**Medium**	**Resolved**
TS-L1	Odds Validation Missing - Allows Invalid American Odds Between -100 and +100	Low	**Resolved**
TS-I1	Zero-Amount Opponent Bets Create Forced Settlement Requirements and Block Creator Withdrawals	**Informational**	**Resolved**
TS-I2	Replace msg! with Anchor Events - Reduce CU Usage and Enable Structured On-Chain Logging	**Informational**	**Resolved**

Detailed Findings

TS-C1 - Opponents Can Settle and Claim Rewards Even If They Lost

Classification

Severity: Critical
Status: Resolved

Description

The *SettleOpponent* instruction currently lacks any validation to check whether the opponent actually won the bet before allowing settlement and reward distribution. As a result, any opponent - including those who bet on the losing side - can call *settle_opponent* and receive a payout from the escrow.

Recommendation

Add a validation step inside *SettleOpponent* to check whether the opponent bet on the winning side before processing any payouts.

Remediation

The vulnerability was fixed in the *f12e0282260bc9ccc9f5c3de3e069adffc4dea32* commit. The admin is responsible for bet settlement.

TS-C2 - Missing Mint Validation in Settlement Allows Griefing and Creator Exploit to Steal Escrowed Tokens

Classification

Severity: Critical
Status: Resolved

Description

The current *settle_bet* and *settle_opponent* instructions accept a user-provided mint account but do not verify that it matches the mint originally used when creating the bet. This introduces two major vulnerabilities:

Attack Scenario 1: Griefing via Wrong Mint

- An attacker can call settle_opponent with a valueless token mint instead of the correct one (e.g., fake USDC).
- This causes rewards to be sent in the wrong token, while real USDC remains locked in the escrow token account.
- Legitimate users are unable to access their rightful winnings.

Attack Scenario 2: Creator Exploit to Steal All Funds

- The creator opens a bet and adds a low-value self-opponent (e.g., 1 token).
- If the creator loses, they settle all opponent bets using a fake mint — so no real tokens are distributed.
- They then settle their own opponent bet last, using the correct mint.
- As the only properly settled opponent, they receive the entire remaining escrow, including all real tokens from others.
- This allows the creator to never lose and steal all opponent funds, even if their side lost.

Recommendation

Store the correct mint in the Bet account at bet creation and validate this mint in all actions regarding Bet.

Remediation

The vulnerability was fixed in the *f12e0282260bc9ccc9f5c3de3e069adffc4dea32* commit by adding a mint whitelisting mechanism.

TS-C3 - Missing Mint Consistency Check Allows Bets with Mismatched or Fake Tokens

Classification

Severity: Critical
Status: Resolved

Description

The current implementation allows users to specify the mint used when placing a bet or joining an existing one. However, there is no enforced consistency check to ensure that all participants use the same mint for the same bet.

This can lead to exploitation where:

- A user creates a bet using a valid token like USDC
- An opponent joins the bet using a valueless or fake token
- The program sees both sides as valid, even though one is using a fake or low-value asset

Recommendation

When the first user creates a bet, store the mint address as part of the bet account. When another user joins a bet, ensure the token mint matches exactly with the one in the bet.

Remediation

The vulnerability was fixed in the *f12e0282260bc9ccc9f5c3de3e069adffc4dea32* commit by verifying bets mint.

TS-C4 - No Access Control on SettleBet Allows Anyone to Arbitrarily Pick a Winner

Classification

Severity: Critical
Status: Resolved

Description

Currently, the *SettleBet* instruction is permissionless, allowing anyone to invoke it and trigger winner selection. However, the winner selection logic is typically sensitive and may rely on off-chain or external decision-making (e.g., oracle result, backend logic, or manual moderation).

Without access control, a malicious actor could:

- Front-run legitimate settlement
- Force early or incorrect settlement
- Exploit timing to manipulate the outcome

Recommendation

Restrict *SettleBet* to be callable only by an authorized signer, stored in the *Master* (global config) account. Validate the signer inside the instruction to ensure only trusted backend/oracle entities can settle the bet.

Remediation

The vulnerability was fixed in the *f12e0282260bc9ccc9f5c3de3e069adffc4dea32* commit by verifying bets mint.

TS-H1 - DoS Risk: init Macro Fails if Malicious User Pre-initializes PDA Token Account

Classification

Severity: High
Status: Resolved

Description

In the current implementation, the escrow account is initialized using the *#[account(init, ...)]* macro.

Since anyone can create associated token accounts (or arbitrary TokenAccounts) for any PDA in Solana, a malicious actor could front-run the next expected PDA using the known seed, and initialize it manually.

If this happens, the *#[account(init)]* macro will fail with an *AccountAlreadyInitialized* error, effectively blocking the creation of the next bet and creating a Denial-of-Service (DoS) condition for the betting system.

Recommendation

There are two viable mitigation strategies:

1. Use *init_if_needed* instead of *init*. This allows the program to continue even if the account already exists - as long as the PDA, mint, and authority match the expected configuration.

2. Create an *Escrow* account in a separate instruction.

Remediation

The vulnerability was fixed in the *f12e0282260bc9ccc9f5c3de3e069adffc4dea32* commit by introducing the *init_if_needed* macro.

TS-H2 - Lack of Token-2022 Extension Validation Can Lead to Transfer Hook Abuse or Fee Miscalculations

Classification

Severity: High
Status: Resolved

Description

The program currently allows any SPL token (including Token-2022 mints) to be used for wagers via *token_interface::transfer_checked*.

However, Token-2022 mints may include dangerous extensions such as:

- Transfer Hook Extension: Enables arbitrary program logic on token transfer (can be malicious).
- Transfer Fee Extension: Automatically deducts a percentage of tokens during transfer.

Without validating or properly handling these extensions, the program is vulnerable to logic inconsistencies and potential abuse.

Example Attack Scenario
Assume a user transfers 100 tokens, but the token has a 5% fee on transfer:

- Only **95** tokens actually land in the bet escrow account.
- But the program still records *bet.wager* as 100.

Recommendation

Restrict allowed extensions to safe types only. If fees on transfer or transfer hook extensions are required, validate the token amount in and token amount out to avoid accounting problems.

Remediation

The vulnerability was fixed in the *f12e0282260bc9ccc9f5c3de3e069adffc4dea32* commit by whitelisting mints that could be used by the protocol.

TS-H3 - Missing Treasury Wallet Validation Allows Fee Redirection by Arbitrary Users

Classification

Severity: High
Status: Resolved

Description

The *process_settle_bet* instruction is designed to be permissionless, which is great for decentralization. However, it currently allows any user to pass in any treasury_wallet address, and fees are sent directly to this provided account without validation.

As a result, a malicious actor can:

- Call *process_settle_bet*
- Provide their own wallet as the *treasury_wallet*
- Steal protocol fees that are meant to go to the real treasury

Recommendation

Store the official treasury wallet address in a global *Config* account. Validate that the provided *treasury_wallet* matches the configured one in every fee-related instruction.

Remediation

The vulnerability was fixed in the *f12e0282260bc9ccc9f5c3de3e069adffc4dea32* commit by verifying the treasury wallet.

TS-M1 - Insecure *update_bet* Instruction Allows Arbitrary Opponent Count Modification

Classification

Severity: Medium
Status: Resolved

Description

The *process_update_bet* instruction is a leftover development utility that allows anyone to arbitrarily update the total_opponents_to_settle field in a *Bet* account.

There is no access control or validation, which means any user can spoof the number of unsettled opponents for a bet.

This introduces a critical issue where:

- A malicious actor can set a non-zero value for *total_opponents_to_settle*
- This prevents the bet creator from closing escrow accounts and withdrawing leftover funds in *settle_opponent*
- The bet stays in a forever-unsettled state, causing fund lockup and unnecessary rent costs

Recommendation

Remove the *update_bet* instruction entirely - it serves no purpose in production.

Remediation

The vulnerability was fixed in the *f12e0282260bc9ccc9f5c3de3e069adffc4dea32* commit by removing the *process_update_bet* instruction.

TS-M2 - Using ATA Authority in Constraints Allows Bet Creator to Block Settlements by Changing Account Authority

Classification

Severity: Medium
Status: Resolved

Description

In the *settle_bet* instruction, the program uses the authority constraint to validate associated token accounts for reward distribution.

However, on Solana, any token account's authority can be changed by the token account owner. This means the bet creator can maliciously transfer the authority of their ATA to another wallet before settlement.

As a result, the *associated_token::authority* constraint will fail, causing the instruction to abort and preventing settlement from being completed. This can be abused to grief the opponent and block reward distribution entirely.

Recommendation

Use any valid token account (not necessarily the ATA) for reward distribution - and validate ownership at runtime. Ideally, separate settlement from account cleanup: allow the bet creator to close accounts and reclaim escrow in a dedicated instruction, so the final opponent isn't responsible for those costs.

Remediation

The vulnerability was fixed in the *f12e0282260bc9ccc9f5c3de3e069adffc4dea32* commit by using a Token Account (not an Associated Token Account) on the opponent settlement.

TS-L1 - Odds Validation Missing – Allows Invalid American Odds Between -100 and +100

Classification

Severity: Low
Status: Resolved

Description

The *calculate_payout* function currently accepts any integer for the odds parameter, including values between -100 and +100. However, in standard American odds formats:

- Positive odds must be strictly greater than +100
- Negative odds must be strictly less than -100

Accepting values like +50, -90, or 0 results in mathematically valid but semantically incorrect calculations. These do not reflect any real-world betting use cases and may lead to misleading payout results.

Recommendation

Add strict validation logic to reject odds in the invalid range. Specifically, disallow any odds between -100 and +100 (inclusive).

Remediation

The vulnerability was fixed in the *f12e0282260bc9ccc9f5c3de3e069adffc4dea32* commit by introducing recommended checks.

TS-I1 - Zero Amount Opponent Bets Create Forced Settlement Requirements and Block Creator Withdrawals

Classification

Severity: Informational
Status: Resolved

Description

The current implementation allows an opponent to place a wager of 0 tokens when joining a bet. Even though this bet has no economic impact, it still increments *total_opponents_to_settle*, and thus must be explicitly settled before the bet creator is allowed to:

- Withdraw remaining funds from the escrow token account
- Close the bet account
- Reclaim rent and unclaimed tokens

Recommendation

Add a validation to reject 0 amount wagers when an opponent joins.

Remediation

The vulnerability was fixed in the *f12e0282260bc9ccc9f5c3de3e069adffc4dea32* commit by introducing a zero amount check.

TS-I2 - Replace msg! with Anchor Events – Reduce CU Usage and Enable Structured On-Chain Logging

Classification

Severity: Informational
Status: Resolved

Description

The program currently uses the *msg!* macro to log key actions like bet creation, settlement, and management actions. While *msg!* is helpful for development-time debugging, it is not suitable for production environments due to two key drawbacks:

- *msg!* logs are not structured and cannot be easily indexed by off-chain systems (like explorers or data services).
- *msg!* statements are expensive in terms of compute units (CU) — each logged line consumes CU, which increases the transaction cost and may push the program over Solana's CU limits in more complex transactions.

Recommendation

Switch to using Anchor's *#[event]* system and the *emit!(...)* macro to log structured, indexable data.

Remediation

The vulnerability was fixed in the *f12e0282260bc9ccc9f5c3de3e069adffc4dea32* commit by introducing Anchor events.

Disclaimer

This report has been prepared in accordance with the current best practices and standards applicable at the time of its preparation. It is intended to provide an analysis and evaluation of the subject matter based on the information available, including any potential vulnerabilities, issues, or risks identified during the assessment process. The scope of this analysis is limited to the data, documents, and materials provided for review, and the conclusions drawn are based on the status of the information at the time of the report.

No representation or warranty, express or implied, is made as to the absolute completeness or accuracy of the information contained in this report. It is important to acknowledge that the findings, conclusions, and recommendations presented are subject to change should there be any modifications to the subject matter. This report should not be viewed as aconclusive or exhaustive evaluation of the subject matter's safety, functionality, or reliability. Stakeholders are encouraged to conduct their own independent reviews, assessments, and validations to ensure the integrity and security of the evaluated subject.

The authors and auditors of this report disclaim any liability for any direct, indirect, incidental, or consequential damages or losses that may result from reliance on this report or its contents. It is the responsibility of the stakeholders to ensure the ongoing monitoring and evaluation of the subject matter to mitigate potential risks or vulnerabilities.

The nature of technology and digital systems means that they are inherently subject to risks, including but not limited to vulnerabilities, bugs, and security threats that may not be foreseeable at the time of this report. The dynamic and evolving nature of technological standards, security practices, and threat landscapes means that absolute security cannot be guaranteed. Despite thorough analysis and evaluation, unforeseen vulnerabilities may exist, and new threats may emerge subsequent to the issuance of this report.

The authors and auditors make no guarantee regarding the impenetrability or infallibility of the subject matter under evaluation. Stakeholders are advised to implement comprehensive security measures, including but not limited to regular updates, patches, and monitoring, to safeguard against potential threats and vulnerabilities.