



# Wager Program Audit Report

---

Prepared by [naman](#)

Version: Draft Report

## Auditors

[naman](#)

[Sanket](#)

Sep 22, 2025

# Contents

---

## 1 About

## 2 Disclaimer

## 3 Risk Classification

## 4 Protocol Summary

## 5 Audit Scope

## 6 Executive Summary

### 6.1 Protocol Info

### 6.2 Issues Found

### 6.3 Findings Summary

## 7 Findings

### 7.1 HIGH RISK

[H-01] Attacker Can Front Run And Hijack Game Session

[H-02] Game Server Can Duplicate Winner Payouts To Winners

### LOW RISK

[L-01] Vault Account Can Be Closed to Reclaim Rent Deposit

[L-02] Missing Toolchain Version in `Anchor.toml`

# 1 About

---

naman (@namx05) is a smart contract security researcher specializing in EVM and Solana ecosystem. He is dedicated to improving blockchain security. He actively researches and explores ways to make protocols safer. He also writes [articles](#) to help developers better understand security.

His past work can be found [here](#). He can be reached on Twitter [@namx05](#) or Telegram [@namx05](#).

## 2 Disclaimer

---

A smart contract security review can't find every vulnerability. It is limited by time, resources, and expertise. The goal is to catch as many issues as possible, but I can't promise complete security. I also can't guarantee that the review will find any problems. To stay safer, it's best to do more reviews, run bug bounty programs, and keep monitoring on-chain activity.

## 3 Risk Classification

---

| Severity           | Impact: High | Impact: Medium | Impact: Low |
|--------------------|--------------|----------------|-------------|
| Likelihood: High   | Critical     | High           | Medium      |
| Likelihood: Medium | High         | Medium         | Low         |
| Likelihood: Low    | Medium       | Low            | Low         |

## 4 Protocol Summary

---

Wager Program is a Solana-based Web3 wagering platform that combines an Anchor smart contract (the `wager-program`) with a TypeScript/Next.js frontend and server utilities to create, manage, and settle wagered game sessions. On-chain logic uses PDAs and a vault to lock SPL token stakes, enforce session state transitions, and perform secure, automated prize distribution; the off-chain components handle session lifecycle, Phantom wallet interactions, Guardian fiat on/off-ramp integration, and test utilities for creating/joining sessions and distributing winnings. The repo includes contract code, integration docs, client utilities, and tests, with security controls for account validation, PDA authority, and token-transfer safety.

## 5 Audit Scope

---

```

1 | wagering-smart-contract/wager-program/src/
2 |   |─ errors.rs
3 |   |─ instructions
4 |   |   |─ create_game_session.rs
5 |   |   |─ distribute_winnings.rs
6 |   |   |─ join_user.rs
7 |   |   |─ mod.rs
8 |   |   |─ pay_to_spawn.rs
9 |   |   |─ record_kill.rs
10 |  |   └─ refund_wager.rs
11 |  |─ lib.rs
12 |  |─ state.rs
13 |  └─ utils.rs

```

## 6 Executive Summary

### 6.1 Protocol Info

|                                  |                         |
|----------------------------------|-------------------------|
| <b>Name:</b>                     | <b>Wager Program</b>    |
| <b>Security Review Timeline:</b> | Sep 20 - Sep 21st, 2025 |
| <b>Repository:</b>               | wagering-smart-contract |
| <b>Methods</b>                   | Manual Review           |

### 6.2 Issues Found

| Severity | Issue Count |
|----------|-------------|
| Critical | 0           |
| High     | 2           |
| Medium   | 0           |
| Low      | 2           |
| Info     | 0           |
| Gas      | 0           |
| Total    | 4           |

## 6.3 Findings Summary

---

| ID   | Title   | Status     |
|------|---|------------|
| H-01 | Attacker Can Front Run And Hijack Game Session        | Unresolved |
| H-02 | Game Server Can Duplicate Winner Payouts To Winners   | Unresolved |
| L-01 | Vault Account Can Be Closed to Reclaim Rent Deposit   | Unresolved |
| L-01 | Missing Toolchain Version in <code>Anchor.toml</code> | Unresolved |

# 7 Findings

## 7.1 HIGH RISK

### [H-01] Attacker Can Front Run And Hijack Game Session

**File(s):** `create_game_session`

**Description:**

The `CreateGameSession` instruction initializes a new `GameSession` account using only `session_id` as a seed (`seeds = [b"game_session", session_id.as_bytes()]`). Since `session_id` is a user-controlled string, any actor observing a pending transaction can submit their own transaction with the same `session_id` but higher fees. This allows them to preemptively create the `GameSession` PDA before the intended game server does. The same design flaw applies to the vault account, which also derives its address from `session_id` only.

The root cause is that the PDA derivation does not include the trusted signer (`game_server.key()`) in its seed, allowing untrusted parties to deterministically predict and occupy addresses.

**Impact:**

An attacker can monitor the network mempool, detect when a legitimate game server is creating a new session, and front run the transaction with the same `session_id`. This allows the attacker to take control of the `game_session` and its associated `vault`, effectively hijacking the game flow, redirecting funds.

**Recommendation(s):**

Include the game server's public key in PDA seeds to bind sessions to their legitimate authority.

```
1 - seeds = [b"game_session", session_id.as_bytes()]
2 + seeds = [b"game_session", session_id.as_bytes(), game_server.key().as_ref()]
3
4 - seeds = [b"vault", session_id.as_bytes()]
5 + seeds = [b"vault", session_id.as_bytes(), game_server.key().as_ref()]
```

**Status:**

Unresolved

**Update from the client:**

### [H-02] Game Server Can Duplicate Winner Payouts To Winners

**File(s):** `distribute_wager.rs`

**Description:**

The `CreateGameSession` instruction initializes a new `GameSession` and its associated vault accounts. Currently, any user can invoke this instruction because the only requirement is that `game_server` is a signer, without verifying that it is an authorized or trusted game server.

This means an arbitrary wallet can act as `game_server` and create fraudulent sessions. Once such a session exists, the attacker can call `distribute_all_winnings_handler`, provide duplicate or manipulated winners in `remaining_accounts`, and drain all bet amounts from the vault.

The root cause is missing access control on game session creation and missing validation that the supplied `remaining_accounts` strictly match the canonical winning team.

**Impact:**

An attacker can bypass intended authority checks, spin up fake game sessions under their control, lure players to deposit into these sessions, and later distribute winnings to themselves or colluding accounts. Combined with the duplicate payout issue, they can steal all funds from the vault.

**Recommendation(s):**

1. Restrict game session creation to a trusted admin authority (e.g., enforce `game_server` to match a predefined `ADMIN_PUBKEY`).
2. Enforce uniqueness of winners in `remaining_accounts` to prevent duplicate payouts.
3. Require `remaining_accounts.len() == 2 * players_per_team` to ensure exact expected entries.
4. Validate that all provided winners are members of the `winning_players` set stored in state.

**Status:**

Unresolved

**Update from the client:**

## LOW RISK

### [L-01] Vault Account Can Be Closed to Reclaim Rent Deposit

**File(s):** `refund_wager`

**Description:**

In the `refund_wager` context, the `vault` account is marked as mut, and while no current instruction allows its closure, the account is susceptible to being closed in the future if not explicitly restricted. Since the rent deposit for the lottery account is typically paid by the lottery creator or admin, best practice dictates that only this party should be permitted to reclaim it.

**Impact:**

The vault creator will lose the rent amount.

**Recommendation(s):**

Close the `vault` account after the refund has been sent and send the rent amount to the payer of the account, i.e. the `game_server`.

**Status:**

Unresolved

**Update from the client:**

### [L-02] Missing Toolchain Version in `Anchor.toml`

**File(s):** `Anchor.toml`

**Description:**

The `Anchor.toml` file lacks explicit `anchor_version` and `solana_version` declarations under the `[toolchain]` section. This omission can lead to version drift across different development environments, potentially introducing unexpected behavior, compilation errors, or inconsistencies during deployment. Without a fixed toolchain version, CI/CD pipelines and team members may inadvertently use incompatible versions of Anchor or Solana.

**Impact:**

Builds and deployments may fail or behave inconsistently across machines, reducing reliability and increasing debugging overhead.

**Recommendation(s):**

Explicitly define the Anchor and Solana versions in the `[toolchain]` section of `Anchor.toml`.

```
1  [toolchain]
2  + anchor_version = "0.31.0"
3  + solana_version = "1.18.25"
```

**Status:**

Unresolved



**Update from the client:**