WAGER PROGRAM SMART CONTRACT AUDIT

1. ABOUT

Naman Jain (@namx05) and Sanket Salavi are security researchers specializing in smart contract and application security across multiple ecosystems, including EVM, Solana, CosmWasm, and TON. Together, they have conducted 200+ audits (35+ in Rust), uncovering 340+ Critical/High severity vulnerabilities across DeFi protocols, staking vaults, NFT platforms, cross-chain systems, governance modules, wallets, and more. With expertise spanning Solidity, Rust, and Func, as well as Web2 penetration testing, they are dedicated to strengthening blockchain security through audits, research, and knowledge sharing.

There past work can be found here:

- Naman
- Sanket

They 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. INTRODUCTION

A time-boxed security review was done by N**aman & Sanket**, with a focus on the security aspects of the smart contracts implementation.
The Wager Program Smart Contract Audit team has been very responsive to Naman's inquiries, demonstrating a strong commitment to security by taking

into account all the recommendations and fixing suggestions from the researchers.

4. RISK CLASSIFICATION

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

4.1 IMPACT

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - leads to a moderate material loss of assets in the protocol or moderately harms a group of users.
- Low - leads to a minor material loss of assets in the protocol or harms a small group of users.

4.2 LIKELIHOOD

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5. EXECUTIVE SUMMARY

5.1 PROTOCOL INFO

Name:	Wager Program
Security Review Timeline:	2 Days
Repository:	wagering-smart-contract/wager-program/src/*
Auditor	Naman, Sanket

5.2 SCOPE

The following smart contracts were in the scope of the security review:

File	nLOC
smart-contracts-refund/wagering-smart-contract/wager-program/src/*	815
Total	815

5.3 FINDINGS COUNT

Severity	Count
Critical	
High	2
Medium	
Low	2
Info	
Gas	
Total	4

5.4 FINDINGS SUMMARY

Overall, the code is well-written. During the review, a total of Four (4) issues were found. Out if which Two (2) were Critical and High issues.

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

ID	Title	Status
L-01	Vault Account Can Be Closed to Reclaim Rent Deposit	Unresolved
L-01	Missing Toolchain Version in `Anchor.toml`	Unresolved

6. FINDINGS

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.

```
 - seeds = [b"game_session", session_id.as_bytes()]
 + seeds = [b"game_session", session_id.as_bytes(), game_server.key().as_
ref()]
```

```
- seeds = [b"vault", session_id.as_bytes()]
+ 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`.

```
[toolchain]
+ anchor_version = "0.31.0"
+ solana_version = "1.18.25"
```

Status: Unresolved

Update from the client: