Solana Program Improvement & Audit for Gaming Protocol

- <u>Document Information</u>

Field	Value
Name	Solana Program Audit
Audited by	AlphaR (https://github.com/AlphaR2)
Project	PrimeSkill Studio
Language	Rust (Anchor Framework)
Date	26/09/2025

- <u>Project Summary</u>

1. Audit Overview

This program was thoroughly audited. The state, instructions and helper logics were properly audited. The program executes player matching, escrow and vault functionality for funds, payouts and anti abuse mechanisms.

2. Key Statistics

- Total Findings: 34 Issues
- Critical: 7 Issues
- High: 9 Issues
- Medium: 13 Issues
- Low: 3 Issues
- Informational: 2 Issues

Note: Critical and High rated issues are issues that must be addressed else the program will encounter runtime errors, Medium issues are more of flawed logic and calculations but these may be the team decision. Low and Informational rated issues are suggestions and findings.

• Bottom Line Assessment

This codebase requires significant refactoring to address fundamental economic and security vulnerabilities before production use

---

- System Overview

Based on the audit analysis, here's the system architecture overview:

Architecture Overview

- **Player Matching:** Backend-controlled centralized matching system where the game server creates sessions and players join specific teams (Team A/B) with manual team selection. These sessions are also created in the program with the session_id
- **Escrow Mechanism:** Individual vault per game session using PDA-based token accounts, where each game holds its session funds as well as payout from that vault. However, critical flaw of using AccountInfo with no state tracking instead of proper vault state management
- **Payout System:** Dual distribution model - winner-takes-all (team-based) or performance-based (kills+spawns formula), with backend authority determining winners and triggering payouts
- **Anti-abuse Measures:** Minimal - basic team size limits and game state checks, but missing duplicate player prevention, spawn limits, and economic exploit protection

Key Components Reviewed

1. **Game Logic** - Single monolithic contract handling all game logic including session creation, player joining, kill recording, spawn management, and fund distribution through multiple instruction handlers

2. **Escrow System** - Individual vault system per game using PDA derivation, but critically flawed with AccountInfo usage instead of structured state tracking, missing deposit verification and balance reconciliation

3. **Matching System** - Team assignment system where the team to be joined is a u8 but a sort of selection range(0 or 1) with basic capacity checking, but this is a flaw because a u8 can contain like 13 or 100 as values. Also missing duplicate player prevention and fair matching algorithms

4. **Payout Contract** - Dual payout mechanisms: fixed-amount winner-takes-all distribution and variable performance-based rewards using confusing kills+spawns formula, both lacking proper vault balance verification

5. **Admin Functions** - Centralized backend authority model where game server wallet controls all operations including game creation, kill recording, and payout distribution, creating single point of failure

---

• Methodology

Audit Approach

1. Manual Code Review - Line-by-line analysis of all logic and flow for the program
2. Attack Vector Analysis - Systematic vulnerability assessment
3. Gas Optimization Review and Byte size optimization - Compute unit efficiency analysis as well as account and struct size to prevent stack overflows
4. Logic Flow Validation - End-to-end game flow testing

Tools Used

- Basic Anchor Cli
- Surfpool for testing

Testing Strategy

- Unit tests for individual functions
- Integration tests for complete game flows
- Edge case and failure mode testing

Summary Table - To see more details about each issue, check the repository below:

- https://github.com/AlphaR2/PrimeSkill-Game-Audit-Report.git
- You can always take the issue ID to see much more details about it plus examples

1. CRITICAL

ID	Title	Severity	Status / Impact
fc-001	Incorrect Data Type Size Calculation	Critical	Failure at runtime
fc-002	Integer Underflow in Spawn System	Critical	Failure when triggered
fc-003	Missing Session ID Length Validation	Critical	If the session id of len > 10 is passed, it won't catch it leading to bad session id
fc-004	Dangerous AccountInfo Usage	Critical	Anchor's in-built type checks are ignored while doing this. Might be exploited
fc-005	Vault Balance Reconciliation Missing	Critical	No verification that total distributed equals vault balance
fc-006	Missing Duplicate Player Check	Critical	Same player can join both teams

fc-007	Code Struct Space Implementation	Critical	Because there is poor space calculations, switch to Anchor InitSpace macro

2. HIGH

ID	Title	Severity	Status / Impact
fh-001	Array Bounds Vulnerability	High	Runtime panics from out-of-bounds access
fh-002	No Spawn Limit Validation	High	Infinite spawn purchases can make games unwinnable
fh-003	Poor Team Index Validation(0 and 1)	High	Invalid team values accepted since it is a u8 if the system feeds in integers other than 0 and 1.
fh-004	No Game State Validation	High	Operations on games in wrong states
fh-005	Remaining Accounts Design Flaw	High	Complex validation, potential manipulation
fh-006	No Authority Validation	High	Unauthorized game operations
fh-007	Double Spend Vulnerability	High	Players could be refunded multiple times
fh-008	Kill Recording logic Missing Validation	High	Invalid kills can be recorded
fh-009	Vault Seed Security Weakness	High	Potential PDA collisions

3. MEDIUM

ID	Title	Severity	Status / Impact
fm-001	Confusing Function Logic	Medium	Incorrect game statistics and developer confusion
fm-002	Fixed Array Size Inefficiency	Medium	128-256 bytes wasted per game in smaller team modes (1v1 and 3v3)
fm-003	Economic Imbalance in Spawn Pricing	Medium	Players get to pay same cost for entry vs additional spawns
fm-004	No Bet Amount Validation	Medium	Lack of amount validation allows for games with 0 bets or extreme amounts
fm-005	Integer Overflow in Kill/Spawn Counters	Medium	Counters wrap to zero after max values
fm-006	Session State Enum Incomplete	Medium	Missing states for cancellation/refunds
fm-007	Winner Validation Logic Error	Medium	Redundant validation code for winners
fm-008	Fixed Amount Distribution Error	Medium	It Doesn't account for actual vault balance, causing left overs or invalid amounts
fm-009	Race Condition in Team Filling	Medium	Multiple players joining simultaneously which might lead to inconsistent states if not checked.
fm-010	Missing Refund State Tracking	Medium	No RefundState account to track who was refunded which might lead to double refunding
fm-011	Data Type Size Optimization	Medium	u16 oversized for expected values in player_spawns and player_kills arrays
fm-012	Redundant Type Casting	Medium	Casting u16 to u16
fm-013	Missing Config Account	Medium	Implement config account and move bet

			amount, the game server pubkey(authority) and spawn numbers there to maintain consistent data and better security

4. LOW

ID	Title	Severity	Status / Impact
fl-001	Excessive Logging	Low	Too many msg! calls for debugging leaded to wasted compute units
fl-002	String vs Array for Session ID	Low	String has length prefix overhead
fl-003	Option<> Usage for Clarity and proper typing of possible None values	Low	Option<Pubkey> which is None at no value is cleaner than Pubkey::default()

5. INFORMATIONAL

ID	Title	Severity	Status / Impact
fin-001	Proper Documentation Needed	Info	
fin-002	Poor Naming Conventions	Info	

Gas Optimization Recommendations

Current Performance Analysis

- **Average compute units per transaction:**
- **Peak compute usage scenarios:**

Optimization Opportunities

1. **Remove Excessive Logging:** Eliminate debug msg! calls saving compute per transaction
2. **Data Type Optimization:** Use u8 instead of u16 for spawns/kills reducing account size and serialization costs
3. **Eliminate Redundant Operations:** Remove unnecessary type casts and duplicate validations
4. **Optimize Account Structure:** Use InitSpace macro for reliable space calculations preventing over-allocation
5. Create and initialize the config account for improved security

---

Recommendations

Immediate Actions Required (Before Launch)

1. **Fix Critical Issues:**

    - FC-001: Correct space calculations to prevent deployment failures

- FC-002: Implement underflow protection to prevent unlimited spawn exploit
- FC-003: Add session ID length validation to prevent invalid IDs
- FC-004: Replace AccountInfo with proper vault state structures
- FC-005: Implement vault balance reconciliation checks
- FC-006: Add duplicate player prevention
- FC-007: Switch to Anchor InitSpace macro for reliable sizing

2. **Address High Severity:**
   - FH-001 through FH-009: Implement proper bounds checking, spawn limits, team validation, and authority checks
3. **Security Enhancements:**
   - Implement comprehensive input validation
   - Add proper error handling and state machine validation
   - Create centralized configuration account for game parameters

Future Improvements

1. **Code Quality:**
   - Implement enum-based team selection for type safety
   - Improve naming conventions throughout codebase
   - Add comprehensive documentation and code comments
   - Establish consistent error handling patterns

2. **Upgrade Mechanisms:**
   - Design upgrade path for program improvements particularly for config and other important accounts
   - Implement emergency pause functionality

Conclusion

Security Assessment

The codebase contains multiple critical vulnerabilities that pose significant risks to user funds and platform integrity. The most severe issues include poor space calculations, integer underflow exploits, economic model flaws, and insufficient validation throughout the system.

Launch Readiness

NOT READY for mainnet deployment. The 7 critical and 9 high severity findings must be resolved before considering production launch. The economic model requires fundamental redesign to prevent exploitation.

Final Recommendations

1. **Priority 1:** Resolve all critical findings immediately - these are deployment blockers
2. **Priority 2:** Address high severity issues to ensure platform security
3. **Priority 3:** Consider architectural improvements for long-term sustainability
4. **Documentation:** Clearly communicate the centralized trust model to users
5. **Testing:** Implement comprehensive testing for economic attack scenarios

Appendix A: Severity Definitions

Severity	Description
Critical	Can lead to loss of funds, complete system compromise, high compute or runtime errors
High	Can lead to program error and major functionality break including high compute and size
Medium	Can lead to unexpected behavior and poor handling of errors
Low	Best practice violations or minor inefficiencies
Informational	Code quality or documentation improvements

Disclaimer

This audit report is based on the code provided. Any changes made after this review may introduce new vulnerabilities not covered in this assessment.

Report prepared by AlphaR (https://github.com/AlphaR2)

Github repo: https://github.com/AlphaR2/PrimeSkill-Game-Audit-Report.git

Contact: @DanielAlphaR2 -/- TG

Date: 26/09/2025