Stonewall
Smart Contract Security

Skinflip Staking Security Audit Report

Auditor: Stonewall Security **Date:** January 2026 **Version:** 1.0 **Repository:** rpajo/solana-staking **Language:** Rust (Anchor Framework) **Chain:** Solana

Executive Summary

This audit reviews a prototype NFT staking smart contract built with the Anchor framework for Solana. The program allows users to stake NFTs into a vault and tracks staking duration for reward calculation.

Findings Summary

Severity	Count
Critical	0
High	1
Medium	2
Low	3
Informational	2

Scope

File	SLOC
programs/skinflip-staking/src/lib.rs	~200

Features Reviewed

- Initialize staking machine
- Stake NFT to vault
- Unstake NFT from vault
- PDA-based stake tracking

Findings

[H-01] Missing Reward Token Distribution Logic

Severity: High **Status:** Open

Description: The contract tracks staking duration via `block.timestamp` but has no mechanism to actually distribute rewards. The `unstake` function calculates elapsed time but doesn't transfer any reward tokens.

Impact:

- Users stake NFTs expecting rewards but receive nothing
- Economic model is incomplete
- No incentive for staking

Recommendation: Implement reward calculation and token distribution:

```
pub fn unstake(ctx: Context<Unstake>) -> Result<()> {
    let staking_account = &ctx.accounts.staking_account;
    let elapsed = Clock::get()?.unix_timestamp - staking_account.staked_at;

    // Calculate rewards based on elapsed time
    let reward_amount = calculate_rewards(elapsed)?;

    // Transfer reward tokens to user
    transfer_rewards(ctx, reward_amount)?;

    // Transfer NFT back to user
    // ...
}
```

[M-01] No Minimum Staking Period Enforcement

Severity: Medium **Status:** Open

Description: Users can stake and immediately unstake their NFTs. The contract calculates elapsed time but doesn't enforce a minimum staking period.

Impact:

- Users can game the system by rapid stake/unstake
- If rewards are implemented, this enables reward farming attacks

Recommendation: Add minimum staking period check:

```
const MIN_STAKING_PERIOD: i64 = 86400; // 24 hours

pub fn unstake(ctx: Context<Unstake>) -> Result<()> {
    let elapsed = Clock::get()?.unix_timestamp - ctx.accounts.staking_account.staked_at
    require!(elapsed >= MIN_STAKING_PERIOD, StakingError::StakingPeriodNotMet);
    // ...
}
```

[M-02] Hardcoded Test Constants in Production Code

Severity: Medium **Status:** Open

Description: The contract contains hardcoded test token mint addresses and authority keys that appear to be environment-specific.

```
// Constants contain test values that shouldn't be in production
const TEST_TOKEN_MINT: Pubkey = ...;
const TEST_AUTHORITY: Pubkey = ...;
```

Impact:

- Deployment confusion between test and production
- Potential use of wrong addresses in production

Recommendation: Use environment-based configuration or initialization parameters rather than hardcoded constants.

[L-01] PDA Seeds Could Include Collection Address

Severity: Low **Status:** Open

Description: PDA seeds use user + NFT + prefix, but don't include the collection address. This could cause issues if multiple collections are supported.

Current:

```
seeds = [b"stake", user.key().as_ref(), nft_mint.key().as_ref()]
```

Recommendation:

```
seeds = [b"stake", collection.key().as_ref(), user.key().as_ref(), nft_mint.key().as_re
```

[L-02] No Event Emission for Off-Chain Indexing

Severity: Low **Status:** Open

Description: The contract uses `msg!()` for logging but doesn't emit structured events that can be easily indexed by off-chain services.

Recommendation: Use Anchor events:

```
#[event]
pub struct StakeEvent {
    pub user: Pubkey,
    pub nft_mint: Pubkey,
    pub timestamp: i64,
}

emit!(StakeEvent { user, nft_mint, timestamp });
```

[L-03] No Admin Controls for Emergency Situations

Severity: Low **Status:** Open

Description: The contract lacks pause functionality or emergency withdrawal mechanisms.

Recommendation: Add pause capability and emergency admin functions.

[I-01] Consider Using Metaplex for NFT Verification

Severity: Informational

Description: The contract doesn't verify NFT metadata or collection membership using Metaplex standards.

[I-02] Prototype Disclaimer Should Be More Prominent

Severity: Informational

Description: The README mentions this is a prototype, but the code itself should include warnings.

Security Features (Positive Findings)

1. **PDA-based tracking**: Uses Program Derived Addresses for stake accounts
2. **Anchor constraints**: Leverages Anchor's account validation
3. **Signer verification**: User must sign stake/unstake transactions

Conclusion

This is a prototype staking contract suitable for learning purposes. Before production use, it requires:

1. Reward distribution implementation
2. Minimum staking period enforcement
3. Production configuration management
4. Emergency pause functionality

Overall Assessment: Not Production Ready (Prototype)