Stonewall
Smart Contract Security

Solana Token Swap AMM Security Audit Report

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

Executive Summary

This audit reviews the Token Swap AMM (Automated Market Maker) example from Solana's official program examples. The contract implements a constant-product AMM with liquidity pools, deposits, withdrawals, and swaps.

Findings Summary

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

Fixed Issues (from previous versions)

Severity	Issue	Status
High	Missing slippage protection in swaps	**Fixed**
Medium	Unchecked arithmetic in fee calculation	**Fixed**

Low	Missing pool existence validation	**Fixed**

Scope

File	SLOC
lib.rs	~60
instructions/*.rs	~400
state/*.rs	~100

Features Reviewed

- Create AMM with fee configuration
- Create liquidity pools
- Deposit liquidity (add tokens)
- Withdraw liquidity (remove tokens)
- Swap tokens with slippage protection

Fixed Findings

[H-01] Missing Slippage Protection in Swaps - **FIXED**

Severity: High **Status:** Fixed

Previous Vulnerable Code:

```
// OLD - No minimum output check
pub fn swap(ctx: Context<Swap>, amount_in: u64) -> Result<()> {
    let amount_out = calculate_output(amount_in, reserve_a, reserve_b);
    transfer_tokens(ctx, amount_out)?;  // User could receive 0 tokens
    Ok(())
}
```

Current Fixed Code:

```
// FIXED - Includes min_output_amount parameter
pub fn swap_exact_tokens_for_tokens(
    ctx: Context<Swap>,
    swap_a: bool,
    input_amount: u64,
    min_output_amount: u64,  // Slippage protection added
) -> Result<()> {
    let output = calculate_output(...);
    require!(output >= min_output_amount, AmmError::SlippageExceeded);
    // ...
}
```

Verification: The current implementation correctly enforces minimum output amounts.

[M-02] Unchecked Arithmetic in Fee Calculation - **FIXED**

Severity: Medium **Status:** Fixed

Previous Issue: Fee calculations used raw arithmetic that could overflow or produce incorrect results.

Current Fix: The code now uses checked math operations and proper basis point calculations with overflow protection.

[L-03] Missing Pool Existence Validation - **FIXED**

Severity: Low **Status:** Fixed

Previous Issue: Operations could be attempted on non-existent pools without proper validation.

Current Fix: Pool accounts are properly validated using Anchor constraints before any operations.

Open Findings

[M-01] First Depositor Can Manipulate Initial Price

Severity: Medium **Status:** Open

Description: The first liquidity provider can set an arbitrary price ratio by choosing any amounts of token A and B. This could be used to front-run expected pools.

Impact:

- Unfair initial pricing
- Potential arbitrage at expense of early LPs

Recommendation: Consider requiring minimum initial liquidity or using an oracle for initial price validation.

[M-02] No Fee Collection Mechanism Visible

Severity: Medium **Status:** Design Question

Description: The AMM initializes with a `fee` parameter, but the mechanism for collecting and distributing fees to LPs isn't clearly visible.

Recommendation: Ensure fees accumulate in the pool reserves and are distributed proportionally to LP token holders on withdrawal.

[L-01] LP Token Decimal Handling

Severity: Low **Status:** Open

Description: When minting LP tokens, the calculation should account for potential decimal mismatches between token A and token B.

[L-02] No Pool Pause Mechanism

Severity: Low **Status:** Open

Description: There's no way to pause a pool in case of emergency or detected exploit.

Recommendation: Add admin-controlled pause functionality.

[I-01] Good Modular Structure

Severity: Informational

Description: The code follows Anchor best practices with separate modules for constants, errors, instructions, and state.

[I-02] Educational Purpose Noted

Severity: Informational

Description: As an official Solana example, this code is primarily educational. Production deployments should add additional safeguards.

Security Features (Positive Findings)

1. **Slippage Protection**: `min_output_amount` parameter prevents sandwich attacks
2. **Anchor Constraints**: Proper account validation using Anchor macros
3. **Separation of Concerns**: Clean modular architecture
4. **Constant Product Formula**: Uses proven x*y=k AMM model

Conclusion

This is a well-structured AMM example that demonstrates proper Solana development patterns. The previous critical issues (slippage protection, arithmetic safety) have been addressed.

Remaining Improvements:

1. First depositor price manipulation mitigation
2. Clear fee distribution mechanism
3. Emergency pause functionality

Overall Assessment: Low Risk (Educational Example)

Stonewall Security *Building Stronger Smart Contracts*