InterFi NETWORK

SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

SOL FORGE AI

INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	Sol Forge AI
Methodology	Automated Analysis, Manual Code Review
Language	Rust
Blockchain	Solana
Centralization	***YES***
Website	https://solforgeai.com/
Telegram	https://t.me/sol_Forge_exchange
X	https://x.com/SolForgeAI
Report Date	March 12, 2025

⚠ Please note: These source codes have not yet been deployed on the Solana main-net and may undergo modifications or alterations prior to their final deployment.

⚠ Verify the authenticity of this report on our website: https://www.github.com/interfinetwork

EXECUTIVE SUMMARY

InterFi has performed the automated and manual analysis of rust codes. Rust codes were reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

Status	Critical 🔴	Major 🟠	Medium 🟡	Minor 🟢	Unknown 🟤
Open	0	0	4	6	0
Acknowledged	0	0	1	0	1
Resolved	0	0	0	0	0
Privileged Functions	`initialize, fund, add_liquidity, remove_liquidity, swap`				

ℹ️ Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.

ℹ️ Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.

TABLE OF CONTENTS

SCOPE OF WORK

InterFi was consulted by Sol Forge AI to conduct the smart contract audit of their rust source codes.

The audit scope of work is strictly limited to mentioned rust file(s) only:

- add_liquidity.rs
- initialize.rs
- mod.rs
- remove_liquidity.rs
- swap.rs
- consts.rs
- errors.rs
- lib.rs
- state.rs
- calc.rs
- mod.rs

⚠ When source codes are not deployed on the main net, they can be modified or altered before main-net deployment.

AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of InterFi's auditing process and methodology:

CONNECT

- The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.

AUDIT

- Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:
  - Solpg Developer Tool
  - Code Analyzer
  - SWC Vulnerabilities Registry
  - DEX Dependencies, e.g., Raydium, Jupiter
- Simulations are performed to identify centralized exploits causing contract and/or trade locks.
- A manual line-by-line analysis is performed to identify contract issues and centralized privileges. We may inspect below mentioned common contract vulnerabilities, and centralized exploits:

Centralized Exploits	○ Token Supply Manipulation ○ Access Control and Authorization ○ Assets Manipulation ○ Ownership Control ○ Liquidity Access ○ Stop and Pause Trading ○ Ownable Library Verification

Common Contract Vulnerabilities	○ Integer Overflow ○ Lack of Arbitrary limits ○ Incorrect Inheritance Order ○ Typographical Errors ○ Requirement Violation ○ Gas Optimization ○ Coding Style Violations ○ Re-entrancy ○ Third-Party Dependencies ○ Potential Sandwich Attacks ○ Irrelevant Codes ○ Divide before multiply ○ Conformance to Rust Naming Guides ○ Compiler Specific Warnings ○ Language Specific Warnings

REPORT

- The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.
- The client's development team reviews the report and makes amendments to rust codes.
- The auditing team provides the final comprehensive report with open and unresolved issues.

PUBLISH

- The client may use the audit report internally or disclose it publicly.

ℹ️ It is important to note that there is no pass or fail in the audit, it is recommended to view the audit as an unbiased assessment of the safety of rust codes.

RISK CATEGORIES

A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized:

Risk Type	Definition
Critical 🔴	These risks pose immediate and severe threats, such as asset theft, data manipulation, or complete loss of contract functionality. They are often easy to exploit and can lead to significant, irreparable damage. Immediate fix is required.
Major 🟠	These risks can significantly impact code performance and security, and they may indirectly lead to asset theft and data loss. They can allow unauthorized access or manipulation of sensitive functions if exploited. Fixing these risks are important.
Medium 🟡	These risks may create attack vectors under certain conditions. They may enable minor unauthorized actions or lead to inefficiencies that can be exploited indirectly to escalate privileges or impact functionality over time.
Minor 🟢	These risks may include inefficiencies, lack of optimizations, code-style violations. These should be addressed to enhance overall code quality and maintainability.
Unknown 🟤	These risks pose uncertain severity to the contract or those who interact with it. Immediate fix is required to mitigate risk uncertainty.

All statuses which are identified in the audit report are categorized here:

Status Type	Definition
Open	Risks are open.
Acknowledged	Risks are acknowledged, but not fixed.
Resolved	Risks are acknowledged and fixed.

CENTRALIZED PRIVILEGES

Centralization risk is the most common cause of cryptography asset loss. When a smart contract has a privileged role, the risk related to centralization is elevated.

There are some well-intended reasons have privileged roles, such as:

- Privileged roles can be granted the power to `pause()` the contract in case of an external attack.
- Privileged roles can use functions like, `include()`, and `exclude()` to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale and to list on an exchange.

Authorizing privileged roles to externally-owned-account (EOA) is dangerous. Lately, centralization-related losses are increasing in frequency and magnitude.

- The client can lower centralization-related risks by implementing below mentioned practices:
- Privileged role's private key must be carefully secured to avoid any potential hack.
- Privileged role should be shared by multi-signature (multi-sig) wallets.
- Authorized privilege can be locked in a contract, user voting, or community DAO can be introduced to unlock the privilege.
- Renouncing the contract ownership, and privileged roles.
- Remove functions with elevated centralization risk.

ℹ️ Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.

AUTOMATED ANALYSIS

Function	Access	Notes
`initialize`	External 🌐	Initializes the program's state by transferring a set amount of lamports from the admin to the global account. Because it's restricted to the `ADMIN` address, the attack surface is small. The main risk is incorrect initial setup or accidental re-initialization if not properly guarded.
`add_liquidity`	External 🌐	Creates a new `LiquidityPool` (if one doesn't exist for the given mint) and transfers tokens/SOL to it. Restricted to `ADMIN` to prevent unauthorized liquidity additions. Potential risks include math errors (overflow/underflow), incorrect pool configuration, or re-initializing an existing pool.
`remove_liquidity`	External 🌐	Removes liquidity from the pool, transferring tokens and SOL out. Also closes the pool's token account and sets `is_migrated = true`. Misuse could drain the pool prematurely or bypass intended checks. Must be carefully used because it finalizes the pool's state (makes it "migrated").
`swap`	External 🌐	Allows token to SOL swaps. Involves reserve manipulation, dynamic pricing, and fee distribution (to admin and creator). Risks include price manipulation if the math is incorrect, potential for front-running, or fee miscalculations. However, the code does apply a 1% fee and a 0.99 "sell reduction."

Internal Trait Methods (in `LiquidityPoolAccount`)

Function	Notes
`remove_liquidity`	Internal implementation invoked by the public `remove_liquidity` instruction. Handles token transfers, SOL transfers, and sets the pool as migrated.
`swap`	Internal `swap` logic, including dynamic pricing, fee calculations, and updating reserves. Called by the public swap instruction.

`add_liquidity_optimized`	Internal logic for adding liquidity to a pool. Called by `public add_liquidity` instruction.
`transfer_token_from_pool`	Helper to safely transfer tokens from the pool's token account (a PDA) to a target account, using the program-derived address as the authority.
`transfer_token_to_pool`	Helper to transfer tokens into the pool's token account from a user's wallet, with the user as the authority.
`transfer_sol_to_pool`	Helper to transfer SOL to the global PDA account, used when adding liquidity or swapping from SOL to tokens.
`transfer_sol_from_pool`	Helper to transfer SOL out of the global PDA account, used when removing liquidity or swapping from tokens to SOL.

MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges	Medium 🟡
CEN-02	Strict Access Control (ADMIN-Only Functions)	

Important privileges are listed below:

```
initialize
fund
add_liquidity
remove_liquidity
swap
```


RECOMMENDATION

Securing private keys or access credentials of deployers, contract owners, operators, and other roles with privileged access is crucial to prevent single points of failure that can compromise contract security.

Use of multi-signature wallets is recommended – These wallets require multiple authorizations to execute sensitive contract functions, reducing the risk associated with single-party control.

Use of decentralized governance model is recommended – This model allows token holders and stakeholders to actively participate in decision-making, such as contract upgrades and parameter adjustments, enhancing overall security and resilience.

ACKNOWLEDGEMENT

Sol Forge AI team argued that centralized and controlled privileges are used as required.

Identifier	Definition	Severity
CEN-03	Seed phrase handling	Minor

Program Derived Addresses (PDAs) rely on seeds to generate unique addresses. The seeds used (`b"global"` or `b"liquidity_pool"`) must be unpredictable or unique enough to avoid potential precomputation attacks. If seeds are guessable, an attacker might attempt to create the same PDA before the legitimate program instruction.

RECOMMENDATION

Use highly unique, unpredictable seed phrases combined with key pieces of state or user-specific data to ensure PDAs are secure against precomputation attacks.

Identifier	Definition	Severity
LOG-01	Fee Distribution Mechanism in 1% Fee Split	Minor

Smart contract correctly computes a 1% fee and splits it 80/20 between the fee recipient `ADMIN` and the pool creator. However, two separate SOL transfers are used to distribute these fees. If one transfer fails, the transaction will revert, but the code flow should ensure both transfers always succeed or the entire swap fails.

RECOMMENDATION

Maintain the current approach but confirm that the fee transfers are either atomic (if one fails, the entire instruction fails) or combine them into a single CPI if feasible. This ensures consistent fee distribution and avoids partial states.

Identifier	Definition	Severity
LOG-03	Use of Floating-Point Arithmetic	Medium 🟡

The contract uses `f64` floating-point arithmetic for critical operations such as calculating fees, swap amounts, and dynamic pricing. Floating-point operations can introduce non-deterministic behavior and rounding errors across different platforms. They are generally discouraged in Solana smart contracts because small precision differences can accumulate and be exploited.

RECOMMENDATION

Adopt a fixed-point arithmetic approach using integer math with a scaling factor (e.g., 1e6 or 1e9) to ensure deterministic results. This approach avoids floating-point rounding issues and aligns better with typical on-chain arithmetic standards.

Identifier	Definition	Severity
LOG-05	Migration Flag `is_migrated` Locks the Pool	Medium 🟡

Once `is_migrated` is set to true `remove_liquidity`, the pool is permanently locked. No further swaps or liquidity operations can occur. This design effectively ends the lifecycle of the pool in a single transaction. If this behavior is intentional (e.g., for a one-time migration or closure), it should be clearly documented.

RECOMMENDATION

Document one-shot behavior of `remove_liquidity`. If a multi-use liquidity removal mechanism is desired, consider implementing a standard share-based approach where users burn pool tokens proportionally, rather than setting a global flag that locks the entire pool.

Identifier	Definition	Severity
LOG-06	Pricing Formula Complexity	Minor

Pricing curve uses logarithms and exponentials `powf, ln` to determine how the price evolves as tokens are bought or sold. While mathematically interesting, it can be challenging to predict all edge cases—especially under extreme input conditions. Small rounding differences might produce unexpected results.

RECOMMENDATION

Perform comprehensive testing of pricing logic under extreme inputs. Document this intended price curve. If possible, use an established AMM formula, like e.g., x*y = k, bonding curves, or stable-swap formulas.

Identifier	Definition	Severity
COD-07	User Input Validation	Minor

Smart contract checks for zero amounts and `minOut` requirements, but additional checks, like time-based deadlines, maximum slippage, are not present. Without these, users can be vulnerable to front-running or price manipulation in certain scenarios.

RECOMMENDATION

Implement optional parameters such as a `deadline` timestamp or a `max_slippage` percentage to protect users from sandwich attacks or volatile price shifts.

Identifier	Definition	Severity
COD-10	Direct and indirect dependencies	Unknown
COD-11	Reliance on Rust language libraries and Anchor framework	
COD-12	Dependence on external data	

Smart contract utilizes various third-party modules and external systems, including but not limited to Rust language libraries, external Anchor framework modules, and web3 applications. Throughout the auditing process, these components are regarded as black boxes with presumed functional correctness. It is crucial to recognize the potential vulnerabilities inherent in these third-party services, which could be compromised or exhibit unpredictable behavior due to changes, such as module upgrades or interface alterations. Such changes might result in increased operational costs, altered contract behavior, or deprecation of previously stable features.

Ensure that any external data, like oracles or cross-program invocations, is validated and handled securely to prevent manipulation.

RECOMMENDATION

Inspect all internal and external dependencies regularly, and push updates as required. Implement interface abstractions or wrappers that can help mitigate issues arising from changes in external contracts.

ACKNOWLEDGEMENT

Sol Forge AI team understands the potential risks from changes such as upgrades or interface alterations. To address this, Sol Forge AI team will regularly monitor all third-party and/or external dependencies and implement updates, when possible, to ensure the stability and security of all smart contracts.

Identifier	Definition	Severity
VOL-01	Unused or Partially Used Custom Errors	Minor

Several custom errors are defined but never appear in the actual logic. This may indicate incomplete or placeholder code.

```
InsufficientShares
InsufficientFunds
InvalidFee
```


RECOMMENDATION

Remove unused errors to reduce code clutter.

Identifier	Definition	Severity
VOL-02	Hard-Coded Constants	Minor

Mentioned constants are hard-coded in the contract:

```
INITIAL_PRICE
PRICE_INCREMENT_STEP
SELL_REDUCTION
GROWTH_FACTOR
```


RECOMMENDATION

Provide an initialization parameter for each constant or store them in an editable on-chain config account for flexibility.

Identifier	Definition	Severity
VOL-03	Potential Logarithm	Medium 🟡

In the swap logic, if the code calculates `avg_price = (max_price - current_price) / (max_price / current_price).ln()`, there is a risk that `(max_price / current_price)` is `1.0` or negative in extreme scenarios, which would make `ln(1.0) = 0 or ln(x <= 0)` invalid. Floating-point can also produce small negative or zero values due to rounding.

RECOMMENDATION

Check for boundary conditions around `(max_price / current_price)` to ensure it never hits `1.0 or 0.`

Identifier	Definition	Severity
VOL-04	Single-Step Liquidity Removal	Medium 🟡

When `remove_liquidity` is called, the contract closes the pool token account and sends all reserves out at once, then sets `is_migrated = true`. There is no partial removal or share-based redemption. This effectively ends the pool in a single step.

RECOMMENDATION

For a standard multi-provider AMM, implement a share-based approach that allows partial liquidity removals and does not forcibly close the entire pool for all participants.

DISCLAIMERS

InterFi Network provides the easy-to-understand audit of rust source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. This audit report could include false positives, false negatives, and other unpredictable results.

CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without InterFi Network's prior written consent.

NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way

to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

FOR AVOIDANCE OF DOUBT, SERVICES, INCLUDING ANY ASSOCIATED AUDIT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

TECHNICAL DISCLAIMER

ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, INTERFI NETWORK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO SERVICES, AUDIT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, INTERFI NETWORK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

WITHOUT LIMITING THE FOREGOING, INTERFI NETWORK MAKES NO WARRANTY OF ANY KIND THAT ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CLIENT'S OR ANY OTHER INDIVIDUAL'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.

TIMELINESS OF CONTENT

The content contained in this audit report is subject to change without any prior notice. InterFi Network does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.

LINKS TO OTHER WEBSITES

This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than InterFi Network. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites' and social accounts' owners. You agree that InterFi Network is not responsible for the content or operation of such websites and social accounts and that InterFi Network shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.

ABOUT INTERFI NETWORK

InterFi Network provides intelligent blockchain solutions. We provide rust development, testing, and auditing services. We have developed 150+ solidity codes, audited 1000+ smart contracts, and analyzed 500,000+ code lines. We have worked on major public blockchains e.g., Ethereum, Binance, Cronos, Doge, Polygon, Avalanche, Metis, Fantom, Bitcoin Cash, Velas, Oasis, etc.

InterFi Network is built by engineers, developers, UI experts, and blockchain enthusiasts. Our team currently consists of 4 core members, and 6+ casual contributors.

Website: https://interfi.network

Email: hello@interfi.network

GitHub: https://github.com/interfinetwork

Telegram (Engineering): https://t.me/interfiaudits

Telegram (Onboarding): https://t.me/interfisupport

interfinetwork

hello@interfi.network

https://interfi.network

SMART CONTRACT AUDITS | SOLIDITY DEVELOPMENT AND TESTING

RELENTLESSLY SECURING PUBLIC AND PRIVATE BLOCKCHAINS