InterFi NETWORK

SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

BIOSOLANA SWAP CONTRACTS

INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	Biosolana Swap
Methodology	Manual Review
Contracts	
Language	Rust
Website	https://www.biokript.com/
Telegram	https://t.me/biokript/
Twitter	https://twitter.com/biokript/
Instagram	https://www.instagram.com/biokript/
Report Date	August 07, 2024
Revision Date	August 19, 2024

ℹ️ 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 source codes. Source 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	0	1	1
Acknowledged	0	1	2	0	1
Resolved	0	1	2	3	0
Noteworthy Privileges	Review PAGE 10 for important centralized and controlled privileges				

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

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 Biokript to conduct the smart contract audit of their solana swap source codes. The audit scope of work is strictly limited to mentioned package(s) only:

- constants.rs
- lib.rs
- create_accounts.rs
- create_pool.rs
- deposit_all_view.rs
- deposit_all.rs
- deposit_single.rs
- router_swap_in_view.rs
- router_swap_in.rs
- router_swap_out_view.rs
- router_swap_out.rs
- swap_exact_in_view.rs
- swap_exact_in.rs
- swap_exact_out_view.rs
- swap_exact_out.rs
- withdraw_all.rs
- withdraw_single.rs
- curve.rs
- config.rs
- create_config.rs
- update_config.rs
- fees.rs
- swap_pair.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:
  - Remix IDE Developer Tool
  - Open Zeppelin Code Analyzer
  - SWC Vulnerabilities Registry
  - DEX Dependencies, e.g., Pancakeswap, Uniswap
- 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 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 source 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 source 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.

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.

MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges	Medium 🟡
CEN-01-01	Security of administrative functionalities	

Some critical operations depend on PDAs or admin-like privileges without a clear audit trail or restriction mechanism visible in the snippets.

Smart contract has administrative functionalities allowing changes to ownership and the fee receiver.

```
set_new_owner
set_new_fee_receiver
```

RECOMMENDATION

Implement role-based access control mechanisms and ensure that operations altering critical state or permissions are logged and traceable.

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

BioKript team commented that centralized control has been designed based on the business requirements of the project. BioKript team will transfer control to company multi-sig to reduce the attack vector.

Identifier	Definition	Severity
LOG-01	Resource exhaustion and denial-of-service	Minor

State Bloat: Contract functions `create_accounts` and `create_pool` can potentially be abused to increase the state stored on-chain excessively. This may lead to increased costs for operating the contract due to the storage fees on Solana.

Network Congestion: Contract functions `deposit_all`, `withdraw_all`, or various swap functions, can be repeatedly called to congest the network, slowing down transaction processing and potentially leading to denial-of-service.

RECOMMENDATION

Implement rate limiting or fees associated with creating resource accounts to mitigate spamming attacks.

Identifier	Definition	Severity
LOG-02	No explicit authentication methods	Major 🟠

Mentioned function does not have sufficient access control:

`create_pool`
`create_accounts`
`deposit_all`
`withdraw_all`
`SwapPair`

Smart contract allows multiple actions, like, deposits, withdrawals, swaps, without clear restrictions on state changes, which can lead to race conditions.

RECOMMENDATION

Use role-based access control or a list of approved addresses that can be modified without needing to deploy a new contract. Ensure that state changes occur after appropriate authentications to prevent unauthorized access.

RESOLUTION

BioKript team commented that there is no need to perform explicit authorization when creating pool. This platform is fully decentralized, and anyone can create a pool without any authorization from anyone. `create_accounts` function is an empty function and we will delate all unnecessary functions. In `deposit_all` function, default slippage percent will be configured when create pool.

Identifier	Definition	Severity
LOG-02-01	No explicit authentication methods	Medium 🟡

`withdraw_all()` and `withdraw_single()` functions need explicit checks to prevent draining liquidity due to unauthorized access or parameter manipulations.

RECOMMENDATION

In `WithdrawAll`, use an advanced rate limiting protocol that dynamically adjusts withdrawal caps based on total pool activity and individual user behaviour. This protocol can include security features to mitigate potential abuse patterns, such as rapid, high-volume withdrawals.

ACKNOWLEDGEMENT

BioKript team has accepted this finding, and kept the code as-is.

Identifier	Definition	Severity
LOG-04	Predictable PDA generation	Major 🟠
LOG-05	Unchecked PDA usage	

If the seeds used to generate PDAs are predictable, attackers can pre-calculate these addresses and potentially exploit them to intercept or misroute transactions, deploy rogue contracts, or access funds and permissions intended for legitimate users. PDAs (Program Derived Addresses) must be validated, and generated securely.

```
impl SwapPair
```

Using `pda: AccountInfo<'info>` with `CHECK` but not actually checking the PDA against expected conditions creates a trust risk. This can allow any account to be passed in without validation.

Using only `token_a` and `token_b` keys as seeds can make PDAs predictable, as these are standard public keys of the tokens involved in the pool. If these seeds are known, an attacker might predict or pre-calculate the PDA before it's officially generated by the legitimate contract operations.

RECOMMENDATION

Implement stricter validation of the `pda` being generated. Ensure PDA is correctly derived with `SEED`.

ACKNOWLEDGEMENT

BioKript team has made some crucial changes by using the seeds derived from the `token_a` and `token_b` keys. However, it is recommended to implement more complexity to the seeds. This could include a nonce, user-specific data, or other elements that cannot be predicted by third parties. This helps ensure that the PDA cannot be pre-calculated by an attacker.

Identifier	Definition	Severity
LOG-06	Unchecked mint authority	Medium 🟡

In pool creation, token accounts, `token_a_for_pda` and `token_b_for_pda`, are initialized with PDA as the authority. There should be strict checks to ensure that the minting rights for these tokens are not susceptible to unauthorized minting attacks, especially since the authority is a PDA.

RECOMMENDATION

Implement stringent validation to ensure that minting rights for these tokens are not manipulated.

RESOLUTION

BioKript team has commented that code is structured to mitigate risks associated with unauthorized minting. All token minting and management operations are controlled by PDAs and validated through stringent constraints and checks.

Identifier	Definition	Severity
LOG-07	Precision loss in mathematical calculations in Swapping and Pool logic	Minor 🟢

Use of floating-point operations or imprecise integer arithmetic can lead to rounding errors or precision loss.

RECOMMENDATION

Utilize fixed-point arithmetic or integer operations that ensure exactness in calculations.

RESOLUTION

BioKript team has used fixed-point arithmetic or integer operations.

Identifier	Definition	Severity
LOG-08	Lack of input validation in Swapping and Pool logic	Medium 🟡

There are several functions where the lack of explicit input validation can lead to logical issues:

```
validate_supply(token_a_amount: u64, token_b_amount: u64) -> Result<()>
pool_tokens_to_trading_tokens(...)
trading_tokens_to_pool_tokens(...)
swap_exact_in(...)
swap_for_exact_out(...)
deposit_single_token_type(...)
withdraw_single_token_type_exact_out(...)
```


RECOMMENDATION

Implement the following input validation strategies across all sensitive functions:

- Range Checks: Ensure all numerical inputs fall within expected and safe ranges.
- Type Checks: Confirm that inputs are of the correct type and format.

RESOLUTION

BioKript team has added parameter range checks as suggested and confirmations that inputs are of correct type. BioKript team has also removed unnecessary view functions.

Identifier	Definition	Severity
COD-02	Lack of documentations and comments	Minor

Smart contract codes lack comprehensive documentation and comments, making it harder to understand.

RECOMMENDATION

Add detailed comments and documentation throughout the codebase. For example,

```
// Parameters:
// - `sender`: A reference to the signer who authorizes the creation of the pool.
public entry fun createPool<CoinType>(sender: &signer) {
    // Register the new pool pairing CoinType with AptosCoin.
    interface::register_pool<CoinType, AptosCoin>(sender);
    // Add liquidity to the pool with equal amounts of CoinType and AptosCoin from
TOTAL_SUPPLY.
    interface::add_liquidity<CoinType, AptosCoin>(sender,
        TOTAL_SUPPLY, TOTAL_SUPPLY, 0, 0
    );
}
```

RESOLUTION

BioKript team will add detailed comments and documentation throughout the codebase

Identifier	Definition	Severity
COD-07	Resource management	Minor

Extracting and depositing coins involve several operations that may fail if all sufficient conditions aren't met, such as insufficient balance. The current logic setup does not account for partial failures or transaction rollbacks explicitly.

RECOMMENDATION

Implement comprehensive error handling and checks to manage partial failures, ensuring that transactions are atomic where necessary.

RESOLUTION

BioKript team will add extra checks, and error handling to atomic transactions.

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

BioKript swap team understands the potential risks from changes such as upgrades or interface alterations. To address this, BioKript 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
COD-13	Note regarding Cross-Program Invocation (CPI) State Dependency	Unknown

If Solana program makes a CPI to another program and expects the state to be unchanged afterwards, there's a risk if the called program alters shared state that the calling program relies on. This isn't re-entrancy, but it's a similar risk where state assumptions are invalidated.

RECOMMENDATION

Any assumptions about the state being consistent after a CPI call should be carefully reviewed. For instance, if a program calls another which then modifies shared resources (like pool tokens or account balances), the original program must not assume these resources remain unchanged.

DISCLAIMERS

InterFi Network provides the easy-to-understand audit of 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 solidity 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