

# SMART CONTRACT AUDIT



interfinetwork



hello@interfi.network



<https://interfi.network>

PREPARED FOR

**LOAN CONTRACT  
DK (DRAGONKING)**



# INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	DragonKing
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	0x18F68b7FC6E21FA4aEe199Ba62701156d091C87e
Blockchain	Binance smart chain
Centralization	Active ownership
Commit	b7975470b9c6ffd456fad3b60d266f53624b5cae
Website	<a href="https://www.dkswap.com">https://www.dkswap.com</a>
Telegram	<a href="https://twitter.com/dragonkingvip9">https://twitter.com/dragonkingvip9</a>
Report Date	January 29, 2023

 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 solidity codes. Solidity codes were reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

Status	Critical <span style="color: red;">●</span>	Major <span style="color: orange;">●</span>	Medium <span style="color: yellow;">●</span>	Minor <span style="color: green;">●</span>	Unknown <span style="color: brown;">●</span>
Open	0	0	1	5	1
Acknowledged	0	0	0	0	0
Partially resolved	0	1	0	0	0
Resolved	0	0	1	4	0
Critical Functions: depositMoney(), pickInterest(), withdrawMoney(), mortgage_DK(), redeem_DK(), setMoonInterest(), setSperiorInvitation(), swapTokenForFund(), authorizeUsdtDK()					

**i** Please note partially resolved issues are acknowledged, but are not fixed in its entirety. They are not fully resistant to exploits, vulnerabilities and/or hacks.

**i** 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.

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



# TABLE OF CONTENTS

TABLE OF CONTENTS .....	4
SCOPE OF WORK.....	5
AUDIT METHODOLOGY .....	6
RISK CATEGORIES .....	8
CENTRALIZED PRIVILEGES .....	9
AUTOMATED ANALYSIS.....	10
MANUAL REVIEW.....	13
OVERALL SCORE .....	29
DISCLAIMERS .....	29
ABOUT INTERFI NETWORK.....	32

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL



## SCOPE OF WORK

InterFi was consulted by Loan to conduct the smart contract audit of their solidity source codes. The audit scope of work is strictly limited to mentioned solidity file(s) only:

- Loan.sol

 If source codes are not deployed on the main net, they can be modified or altered before main-net deployment. Verify the contract's deployment status below:

Public Contract Link	
<a href="https://bscscan.com/address/0x18F68b7FC6E21FA4aEe199Ba62701156d091C87e#code">https://bscscan.com/address/0x18F68b7FC6E21FA4aEe199Ba62701156d091C87e#code</a>	
Contract Name	Loan
Compiler Version	0.8.7
License	GPL-3.0



# 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	<ul style="list-style-type: none"><li>○ Token Supply Manipulation</li><li>○ Access Control and Authorization</li><li>○ Assets Manipulation</li><li>○ Ownership Control</li><li>○ Liquidity Access</li><li>○ Stop and Pause Trading</li><li>○ Ownable Library Verification</li></ul>
----------------------	---



## 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 Solidity 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 solidity 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.

 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 solidity codes.



## RISK CATEGORIES

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. 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 here for the reader to review:

Risk Type	Definition
Critical 	These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
Major 	These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.
Medium 	These risks should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy-related vulnerabilities should be fixed to deter exploits.
Minor 	These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.
Unknown 	These risks pose uncertain severity to the contract or those who interact with it. They should be fixed immediately to mitigate the risk uncertainty.

All statuses which are identified in the audit report are categorized here for the reader to review:

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.



## AUTOMATED ANALYSIS

Symbol	Definition
	Function modifies state
	Function is payable
	Function is internal
	Function is private
!	Function is important

```

| **IERC20** | Interface | |||
| L | totalSupply | External ! | |NO ! |
| L | balanceOf | External ! | |NO ! |
| L | transfer | External ! |  |NO ! |
| L | allowance | External ! | |NO ! |
| L | approve | External ! |  |NO ! |
| L | transferFrom | External ! |  |NO ! |
|||||
| **ISwapRouter** | Interface | |||
| L | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ! |  |NO ! |
|||||
| **Price** | Interface | |||
| L | changePrice | External ! |  |NO ! |
| L | getPrice | External ! | |NO ! |
|||||
| **Loan** | Interface | |||
| L | depositMoney | External ! |  |NO ! |
| L | calculate | External ! | |NO ! |
| L | pickInterest | External ! |  |NO ! |

```



```

| L | withdrawMoney | External ! | ● | NO ! |
| L | getDkAndUsdtQuantity | External ! | | NO ! |
| L | mortgage_DK | External ! | ● | NO ! |
| L | redeem_DK | External ! | ● | NO ! |
| L | getLoanUserInformation | External ! | | NO ! |
| L | getLoanInterest | External ! | | NO ! |
| L | getPrice | External ! | | NO ! |
| L | takeOutToken | External ! | ● | NO ! |
| L | setMoonInterest | External ! | ● | NO ! |
| L | setSuperiorInvitation | External ! | ● | NO ! |
| L | getSuperior | External ! | | NO ! |
| L | getTotalRebateIncome | External ! | | NO ! |
| L | getSuperiorInformation | External ! | | NO ! |
| L | getLunarization | External ! | | NO ! |
| L | receiveTotalRebateIncome | External ! | ● | NO ! |
| L | getinviteArray | External ! | | NO ! |

```

```

|||||

```

```

| **Loan** | Implementation | |||
| L | depositMoney | External ! | ● | withdraw |
| L | calculate | External ! | | NO ! |
| L | _calculateInterest | Internal 🔒 | | |
| L | pickInterest | External ! | ● | withdraw |
| L | _pickInterest | Internal 🔒 | ● | |
| L | withdrawMoney | External ! | ● | withdraw |
| L | getDkAndUsdtQuantity | External ! | | NO ! |
| L | _getDkAndUsdtQuantity | Internal 🔒 | | |
| L | mortgage_DK | External ! | ● | withdraw |
| L | redeem_DK | External ! | ● | withdraw |

```

INTERFI  
CONFIDENTIAL



	Ⓛ		getLoanUserInformation		External	!		NO!	
	Ⓛ		getLoanInterest		External	!		NO!	
	Ⓛ		_getLoanInterest		Internal	🔒			
	Ⓛ		buyUsdtSpendDk		Internal	🔒			
	Ⓛ		buyDkSpendUsdt		Internal	🔒			
	Ⓛ		_getInviterQuantity		Internal	🔒		🔴	
	Ⓛ		setMoonInterest		External	!		🔴  NO!	
	Ⓛ		getSuperiorInformation		External	!		NO!	
	Ⓛ		receiveTotalRebateIncome		Public	!		🔴  NO!	
	Ⓛ		setSperiorInvitation		External	!		🔴  NO!	
	Ⓛ		getSperior		External	!		NO!	
	Ⓛ		getinviteArray		External	!		NO!	
	Ⓛ		_getRebateIncome		Internal	🔒			
	Ⓛ		getTotalRebateIncome		Public	!		NO!	
	Ⓛ		getLunarization		External	!		NO!	
	Ⓛ		_getInterestRate		Internal	🔒		🔴	
	Ⓛ		getPrice		External	!		NO!	
	Ⓛ		swapTokenForFund		Internal	🔒		🔴	
	Ⓛ		authorizeUsdtDK		External	!		🔴  NO!	

TERFI  
CONFIDENTIALINTERFI  
CONFIDENTIAL

## MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Missing role-based access control	Major 🟡

Below mentioned functions must be provided adequate access control to deter unauthorized access:

`authorizeUsdtDK()` – This function sets unlimited allowances. Implement role-based access control.

`setMoonInterest()` – `manageraddress` can access this function.

### RECOMMENDATION

Implement a role-based access control or an ownership model to restrict access to sensitive functions.

### PARTIAL RESOLUTION\*

`setMoonInterest()` function is modified with more `require` statements to allow value change within certain range.

### AUDIT COMMENT

`authorizeUsdtDK()` function is missing adequate role-based access control.



Identifier	Definition	Severity
LOG-01	Inadequate input validation	Medium 🟡

Below mentioned functions must be provided adequate require statements to validate inputs:

```
setMoonInterest()  
receiveTotalRebateIncome()  
setSperiorInvitation()
```

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL


## RECOMMENDATION

Validate external contract addresses and input parameters to avoid interactions with malicious contracts or erroneous inputs.

## RESOLUTION

Project team has modified aforementioned functions with require statements to validate inputs.



Identifier	Definition	Severity
LOG-02	Potential front-running	Minor 

Potential front-running also classified as – sandwich attack happens when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by front-running a transaction to purchase assets and make profits by back-running a transaction to sell assets. Below mentioned functions are called without setting restrictions on slippage or minimum output:


```
swapExactTokensForTokensSupportingFeeOnTransferTokens()
```

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

## RECOMMENDATION

Consider implementing mechanisms to mitigate front-running, such as using commit-reveal schemes or adding slippage protection.



Identifier	Definition	Severity
LOG-03	Re-entrancy	Critical 

withdraw modifier is used to prevent re-entrancy attacks. However, it is not implemented correctly. Correct way is to set the lock before the function body and reset it after. Current implementation does not prevent re-entrant calls within the same transaction.

Below mentioned functions are used without appropriate re-entrancy guard:

```
depositMoney()
pickInterest()
withdrawMoney()
mortgage_DK()
redeem_DK()
swapTokenForFund()
```

#### RECOMMENDATION

Modify withdraw modifier to correctly implement the re-entrancy guard. Use standard *OpenZeppelin* ReentrancyGuard library.

#### PARTIAL RESOLUTION\*

Project team has modified withdraw modifier, and added custom re-entrancy guard.

#### AUDIT COMMENT

*OpenZeppelin* nonReentrant modifier is a part of a well-tested and widely-used library. It is recommended to use *OpenZeppelin* nonReentrant modifier to re-entrancy attacks.

```
enum ReentrancyGuard {
    NotEntered,
    Entered
}
```





```
ReentrancyGuard private _status;

modifier nonReentrant() {
    require(_status != ReentrancyGuard.Entered, "Reentrant call");
    _status = ReentrancyGuard.Entered;
    _;
    _status = ReentrancyGuard.NotEntered;
}
```

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL



Identifier	Definition	Severity
LOG-04	Note regarding flash loan attacks	Medium 🟡
LOA-01	Dependence on external price feeds	

Smart contract does not directly interact with flash loans. However, since it interfaces with ISwapRouter, smart contract might be exposed to price manipulation if swap occurs in a pool with low liquidity. Price manipulation can lead to unfavorable conditions for functions that depend on the token's price.


Smart contract depends on external Price contract for pricing information. If this feed is manipulated, it may impact the contract's functions.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

## RECOMMENDATION

Implement additional checks or oracles to ensure price integrity. Consider using multiple price feeds or time-weighted average prices to mitigate this risk.



Identifier	Definition	Severity
COD-01	Block gas limit	Minor 

Below mentioned functions may iterate over potentially large arrays or mappings over time, which may hit block gas limit, causing transactions to fail:

```
receiveTotalRebateIncome  
setSperiorInvitation()
```

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

## RECOMMENDATION

Implement gas-efficient patterns, like paginated functions, or limit the size of iterable state variables.



Identifier	Definition	Severity
COD-02	Timestamp manipulation via <code>block.timestamp</code> and <code>block.number</code>	Minor 


Be aware that the timestamp of the block can be manipulated by a miner. When the contract uses the timestamp to seed a random number, the miner can actually post a timestamp within 15 seconds of the block being validated, effectively allowing the miner to precompute an option more favorable to their chances.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

## RECOMMENDATION

To maintain block integrity, follow 15 seconds rule, and scale time dependent events accordingly.



Identifier	Definition	Severity
COD-06	Hardcoded addresses	Minor 

Mentioned 0x addresses are found in the contract.

manageraddress 0xce063ad1c88EE864d1Aa818be98C5D020540e405

DK 0xB5f6e4236591D1e68d93A40E24Ebfe9E7CEe7F7b

Price 0x55b028f53B758D60c5f9E261655186b9fcE295A7

Cont 0xB5f6e4236591D1e68d93A40E24Ebfe9E7CEe7F7b

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

## RECOMMENDATION

Private keys of externally owned accounts must be secured carefully. Only interact with trusted addresses and contracts.



Identifier	Definition	Severity
COD-10	Direct and indirect dependencies	Unknown 🟡
LOA-02	Arbitrary external contract calls	

Smart contract is interacting with third party protocols e.g., Market Makers, External Contracts, Web 3 Applications, Open Zeppelin tools. The scope of the audit treats these entities as black boxes and assumes their functional correctness. However, in the real world, all of them can be compromised, and exploited. Moreover, upgrades in these entities can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

Smart contract interacts with external contracts, e.g., ISwapRouter, IERC20, Price. There are no checks to ensure that these contracts are trusted.


Comment: ISwapRouter, usdt contracts are widely used on BSC chain, and can be considered trusted.

ISwapRouter 0x10ED43C718714eb63d5aA57B78B54704E256024E  
usdt 0x55d398326f99059fF775485246999027B3197955

## RECOMMENDATION

Inspect third party dependencies regularly, and mitigate severe impacts whenever necessary.



Identifier	Definition	Severity
COD-11	Unchecked return values	Minor 

Smart contract performs token transfers without checking the return values. This is risky if smart contract is interacting with non-standard or malicious token contracts.

```
USDT.transfer(...)  
DK_TOKEN.transfer(...)  
USDT.transferFrom(...)  
token.transfer(...)
```

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

## RECOMMENDATION


Ensure that token transfer calls return a boolean success value and check it:

```
require(USDT.transfer(...), "Transfer failed");
```

## RESOLUTION

Project team has added return value checks in aforementioned functions.



Identifier	Definition	Severity
COD-12	Lack of event-driven architecture	Minor 

Smart contract does not emit events for critical state changes like deposit, withdrawal, interest pick-up, or changes in configuration. This makes tracking and verifying transactions on-chain difficult.

## RECOMMENDATION


Events improve transparency and provide a more granular view of contract activity. Implement and emit events for critical operations.

## RESOLUTION

Project team has added events for critical state changes like deposit, withdrawal, interest pick-up.





Identifier	Definition	Severity
VOL-02	Typographical Error	Minor 

Typographical errors are found in:

setSperiorInvitation()

getSperior()

INTERFI  
CONFIDENTIAL

INTERFI  
CONFIDENTIAL

## RECOMMENDATION

Fix typographical errors.



Identifier	Definition	Severity
VOL-03	Integer overflow and underflow	Minor ●


Smart contract does not use SafeMath for arithmetic operations, which may lead to integer overflows or underflows.

TERFI  
CONFIDENTIALINTERFI  
CONFIDENTIAL

## RECOMMENDATION

Use SafeMath library for arithmetic operations.



Identifier	Definition	Severity
COM-01	Floating compiler status	Minor 

Compiler is set to ^0.8.0

INTERFI  
CONFIDENTIAL

INTERFI  
CONFIDENTIAL

## RECOMMENDATION

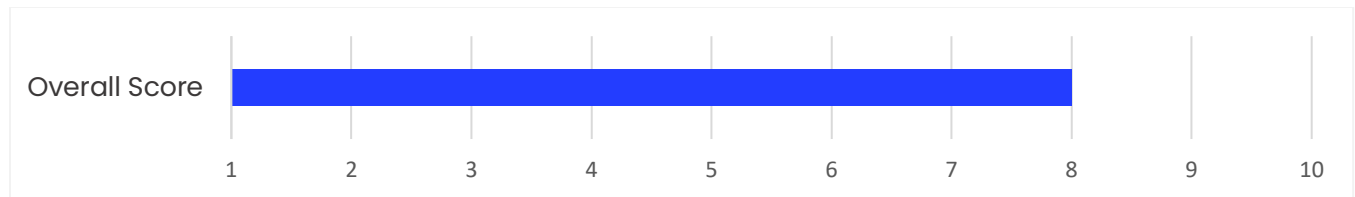
Pragma should be fixed to the version that you're indenting to deploy your contracts with.



## OVERALL SCORE

Source code of loan smart contract has completed solidity audit and received score of **8.0**.

Users, developers, and stakeholders should not solely rely on the audit score as a guarantee of contract's security or performance. Instead, they should view the score as one component of a comprehensive code evaluation process.



INTERFI  
CONFIDENTIAL

INTERFI  
CONFIDENTIAL



## DISCLAIMERS

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

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL



## 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](mailto: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