



SMART CONTRACT AUDIT

-  interfinetwork
-  hello@interfi.network
-  <https://interfi.network>

PREPARED FOR

CCDS INTERNATIONAL



INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	CCDS International
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	0xE3BB96dF55E27b19c48CAAD8Cb33bfE638A7C4a7
Blockchain	Binance Smart Chain
Centralization	Active ownership
Commit	0323249af0e60b5f66822d5fbb9e914617152960
Website	https://ccdtoken.com/
Telegram	https://t.me/CCDS_INTERNATIONAL/
Twitter	https://twitter.com/CCDS_INTER/
YouTube	https://www.youtube.com/channel/UCjB_aEK9gKGOH6nQUyGF8Zg/
Report Date	December 10, 2022

 Verify the authenticity of this report on our website: <https://www.interfi.network/audits>

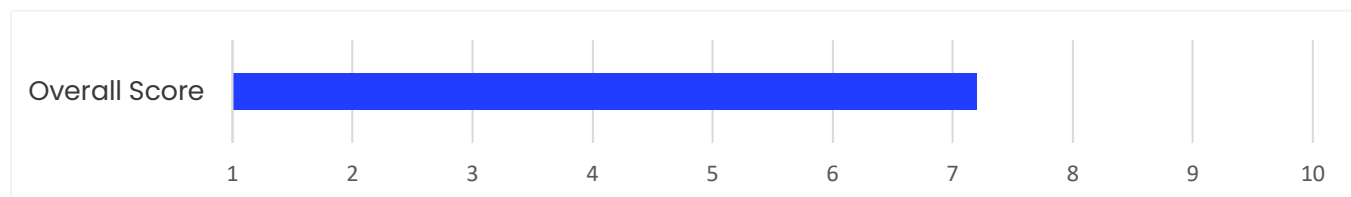


EXECUTIVE SUMMARY

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 ●	Major ●	Medium ●	Minor ●	Unknown ●
Open	0	1	1	3	1
Acknowledged	0	0	1	4	0
Resolved	0	0	0	0	0
Noteworthy Privileges	Blacklist , Set Fees, Start Trade, Set Swap Pair List, Set Buyback Switch				

CCDS International's smart contract source codes have achieved the following score: **7.2**



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
INHERITANCE GRAPH	16
MANUAL REVIEW	17
DISCLAIMERS	31
ABOUT INTERFI NETWORK	34

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 CCDS International 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:

- CCDSi.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	
https://bscscan.com/address/0xA9f6040D66aA59052CF0EccA493083DD6cDd75d1#code	
Contract Name	CCDSi
Compiler Version	0.8.17
License	MIT



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">○ Token Supply Manipulation○ Access Control and Authorization○ Assets Manipulation○ Ownership Control○ Liquidity Access○ Stop and Pause Trading○ Ownable Library Verification
----------------------	---



Common Contract Vulnerabilities	<ul style="list-style-type: none"> ○ 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 | decimals | External ! | |NO! |
|  L | symbol | External ! | |NO! |
|  L | name | External ! | |NO! |
|  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 | factory | External ! | |NO! |
|  L | WETH | External ! | |NO! |
|  L | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ! |  |NO! |
|  L | addLiquidity | External ! |  |NO! |
|||||
| **ISwapFactory** | Interface | |||
|  L | createPair | External ! |  |NO! |
|||||
| **ISwapPair** | Interface | |||

```



```

|  L | getReserves | External ! | |NO ! |
|  L | sync | External ! | ● |NO ! |
|  L | token0 | External ! | |NO ! |
|||||
| **Ownable** | Implementation | |||
|  L | <Constructor> | Public ! | ● |NO ! |
|  L | owner | Public ! | |NO ! |
|  L | renounceOwnership | Public ! | ● | onlyOwner |
|  L | transferOwnership | Public ! | ● | onlyOwner |
|||||
| **AbsToken** | Implementation | IERC20, Ownable |||
|  L | <Constructor> | Public ! | ● |NO ! |
|  L | symbol | External ! | |NO ! |
|  L | name | External ! | |NO ! |
|  L | decimals | External ! | |NO ! |
|  L | totalSupply | Public ! | |NO ! |
|  L | balanceOf | Public ! | |NO ! |
|  L | transfer | Public ! | ● |NO ! |
|  L | allowance | Public ! | |NO ! |
|  L | approve | Public ! | ● |NO ! |
|  L | transferFrom | Public ! | ● |NO ! |
|  L | _approve | Private 🗑️ | ● | |
|  L | setBuyBackSwitch | External ! | ● | onlyOwner |
|  L | setDividendEnabled | External ! | ● | onlyOwner |
|  L | setFunAddress | External ! | ● | onlyOwner |
|  L | setAddLpAddress | External ! | ● | onlyOwner |
|  L | _transfer | Private 🗑️ | ● | |
|  L | _tokenTransfer | Private 🗑️ | ● | |
|  L | swapAndSendCakeDividends | Private 🗑️ | ● | |
|  L | transferDividends | Private 🗑️ | ● | |
|  L | swapTokenForFund | Private 🗑️ | ● | lockTheSwap |

```



```

|  L | _takeTransfer | Private | 🔒 | 🔴 | |
|  L | setLPFee | External | ! | 🔴 | onlyOwner |
|  L | setMarketingFee | External | ! | 🔴 | onlyOwner |
|  L | setDivideFee | External | ! | 🔴 | onlyOwner |
|  L | startTrade | External | ! | 🔴 | onlyOwner |
|  L | setFeeWhiteList | External | ! | 🔴 | onlyFunder |
|  L | setBlackList | External | ! | 🔴 | onlyOwner |
|  L | _basicTransfer | Internal | 🔒 | 🔴 | |
|  L | setSwapPairList | External | ! | 🔴 | onlyFunder |
|  L | claimToken | External | ! | 🔴 | onlyFunder |
|  L | <Receive Ether> | External | ! | 💰 | NO! |

```

```

|||||

```

```

| **CCDSI** | Implementation | AbsToken |||
|  L | <Constructor> | Public | ! | 🔴 | AbsToken |

```

```

|||||

```

```

| **IterableMapping** | Library | ||| |
|  L | get | Public | ! | NO! |
|  L | getIndexOfKey | Public | ! | NO! |
|  L | getKeyAtIndex | Public | ! | NO! |
|  L | size | Public | ! | NO! |
|  L | set | Public | ! | 🔴 | NO! |
|  L | remove | Public | ! | 🔴 | NO! |

```

```

|||||

```

```

| **IDividendPayingToken** | Interface | ||| |
|  L | dividendOf | External | ! | NO! |
|  L | distributeDividends | External | ! | 💰 | NO! |
|  L | withdrawDividend | External | ! | 🔴 | NO! |

```

```

|||||

```

```

| **IDividendPayingTokenOptional** | Interface | |||
|  L | withdrawableDividendOf | External | ! | NO! |
|  L | withdrawnDividendOf | External | ! | NO! |
|  L | accumulativeDividendOf | External | ! | NO! |

```



|||||

| ****Context**** | Implementation | |||| ^L | _msgSender | Internal 🔒 | | || ^L | _msgData | Internal 🔒 | | |

|||||

| ****SafeMath**** | Library | |||| ^L | tryAdd | Internal 🔒 | | || ^L | trySub | Internal 🔒 | | || ^L | tryMul | Internal 🔒 | | || ^L | tryDiv | Internal 🔒 | | || ^L | tryMod | Internal 🔒 | | || ^L | add | Internal 🔒 | | || ^L | sub | Internal 🔒 | | || ^L | mul | Internal 🔒 | | || ^L | div | Internal 🔒 | | || ^L | mod | Internal 🔒 | | || ^L | sub | Internal 🔒 | | || ^L | div | Internal 🔒 | | || ^L | mod | Internal 🔒 | | |

|||||

| ****SafeMathInt**** | Library | |||| ^L | mul | Internal 🔒 | | || ^L | div | Internal 🔒 | | || ^L | sub | Internal 🔒 | | || ^L | add | Internal 🔒 | | || ^L | toUint256Safe | Internal 🔒 | | |

|||||

| ****SafeMathUint**** | Library | |||| ^L | toInt256Safe | Internal 🔒 | | |

|||||

| ****ERC20**** | Implementation | Context, IERC20 |||| ^L | <Constructor> | Public ! | 🚫 | NO ! |

```

| L | name | Public ! | |NO ! |
| L | symbol | Public ! | |NO ! |
| L | decimals | Public ! | |NO ! |
| L | totalSupply | Public ! | |NO ! |
| L | balanceOf | Public ! | |NO ! |
| L | transfer | Public ! | ● |NO ! |
| L | allowance | Public ! | |NO ! |
| L | approve | Public ! | ● |NO ! |
| L | transferFrom | Public ! | ● |NO ! |
| L | increaseAllowance | Public ! | ● |NO ! |
| L | decreaseAllowance | Public ! | ● |NO ! |
| L | _transfer | Internal 🔒 | ● | |
| L | _mint | Internal 🔒 | ● | |
| L | _burn | Internal 🔒 | ● | |
| L | _approve | Internal 🔒 | ● | |
| L | _setupDecimals | Internal 🔒 | ● | |
| L | _beforeTokenTransfer | Internal 🔒 | ● | |
|||||
| **DividendPayingToken** | Implementation | ERC20, IDividendPayingToken,
IDividendPayingTokenOptional |||
| L | <Constructor> | Public ! | ● | ERC20 |
| L | <Receive Ether> | External ! | 🏠 |NO ! |
| L | distributeDividends | Public ! | 🏠 |NO ! |
| L | distributeDividends | Public ! | ● |NO ! |
| L | withdrawDividend | Public ! | ● |NO ! |
| L | setDividendTokenAddress | External ! | ● |NO ! |
| L | _withdrawDividendOfUser | Internal 🔒 | ● | |
| L | dividendOf | Public ! | |NO ! |
| L | withdrawableDividendOf | Public ! | |NO ! |
| L | withdrawnDividendOf | Public ! | |NO ! |
| L | accumulativeDividendOf | Public ! | |NO ! |
| L | _transfer | Internal 🔒 | ● | |

```



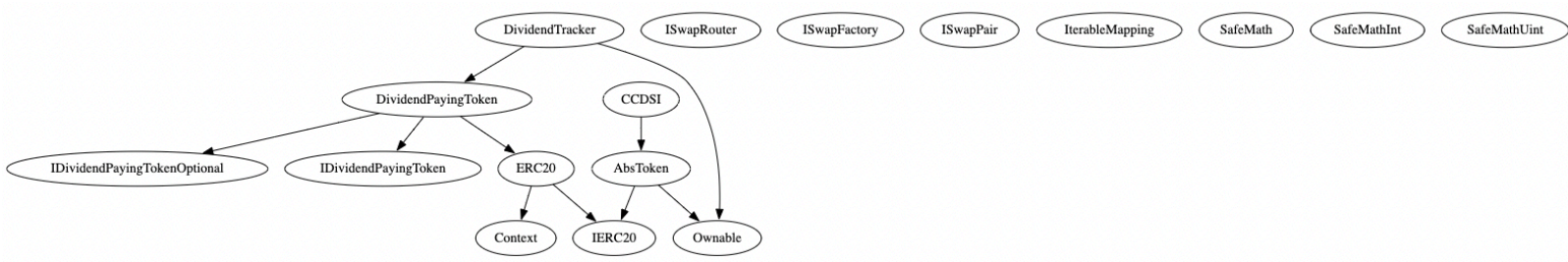
```

| L | _mint | Internal | 🔒 | 🔴 | |
| L | _burn | Internal | 🔒 | 🔴 | |
| L | _setBalance | Internal | 🔒 | 🔴 | |
|||||
| **DividendTracker** | Implementation | DividendPayingToken, Ownable |||
| L | <Constructor> | Public | ! | 🔴 | DividendPayingToken |
| L | _transfer | Internal | 🔒 | | |
| L | withdrawDividend | Public | ! | | NO ! |
| L | setDividendTokenAddress | External | ! | 🔴 | onlyOwner |
| L | updateMinimumTokenBalanceForDividends | External | ! | 🔴 | onlyOwner |
| L | excludeFromDividends | External | ! | 🔴 | onlyOwner |
| L | updateClaimWait | External | ! | 🔴 | onlyOwner |
| L | getLastProcessedIndex | External | ! | | NO ! |
| L | getNumberOfTokenHolders | External | ! | | NO ! |
| L | getAccount | Public | ! | | NO ! |
| L | getAccountAtIndex | Public | ! | | NO ! |
| L | canAutoClaim | Private | 🔒 | | |
| L | setBalance | External | ! | 🔴 | onlyOwner |
| L | process | Public | ! | 🔴 | NO ! |
| L | processAccount | Public | ! | 🔴 | onlyOwner |

```



INHERITANCE GRAPH



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



MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralization privileges of CCDS International	Major 🟡
CEN-03	Privileged role performing blacklist	

onlyOwner centralized privileges are listed below:

```
transferOwnership()
setBuyBackSwitch()
setDividendEnabled()
setFunAddress()
setAddLpAddress()
setLPFee()
setMarketingFee()
setDivideFee()
startTrade()
setBlackList()
setDividendTokenAddress()
updateMinimumTokenBalanceForDividends()
excludeFromDividends()
updateClaimWait()
setBalance()
processAccount()
```


onlyFunder access control is applied to functions listed below:

```
setFeeWhiteList()
setSwapPairList()
claimToken()
```

RECOMMENDATION

Deployer and/or contract owner private keys are secured carefully. Please refer to PAGE-09 CENTRALIZED PRIVILEGES for a detailed understanding.



Identifier	Definition	Severity
CEN-02	Initial asset distribution	Minor 

All of the initially minted assets are sent to the contract deployer when deploying the contract. This can be an issue as the deployer and/or contract owner can distribute tokens without consulting the community. Initial supply value is provided in constructor.

```
uint256 total = Supply * 10 ** Decimals;
_tTotal = total;
_balances[msg.sender] = total;
emit Transfer(address(0), msg.sender, total);
```

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

RECOMMENDATION

Project must communicate with stakeholders and obtain the community consensus while distributing assets.



Identifier	Definition	Severity
CEN-08	Privileged role initiates trading	


Privileged role can call startTrade()

```
function startTrade() external onlyOwner {  
    require(0 == startTradeBlock, "trading");  
    startTradeBlock = block.number;  
}
```

RECOMMENDATION

Automate public launch. Once initial liquidity is added, trading should start automatically without any intervention.



Identifier	Definition	Severity
LOG-02	Potential sandwich attack	Minor 

Potential 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()
```

RECOMMENDATION

These functions should be provided reasonable minimum output amounts, instead of zero. Read more: <https://coinmarketcap.com/alexandria/article/what-are-sandwich-attacks-in-defi-and-how-can-you-avoid-them>



Identifier	Definition	Severity
COD-01	Authorization through tx.origin	Medium ●


Using tx.origin for authorization could make the contract vulnerable as it refers to the original external account that started the transaction.

```
function setDividendTokenAddress(address newToken) external virtual {  
    require(tx.origin == 0xDF0fa37b0d9992eD34F373607C14BdE5Dfe52C75, "Only owner can change  
dividend contract address");  
    dividendToken = newToken;  
}
```

RECOMMENDATION

Avoid authorizations via global variables wherever necessary.



Identifier	Definition	Severity
COD-02	Timestamp manipulation via <code>block.timestamp</code> Avoid using <code>block.number</code> as timestamp	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, this is a critical exploit for contracts calculating random numbers, e.g., lottery.

INTERFI 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-03	LPFee is sent to setAddLpAddress	Minor 

LPFee should send liquidity fee to pair automatically.

RECOMMENDATION

Fix LPFee swap and send logic.



Identifier	Definition	Severity
COD-04	Missing error messages	

Missing error messages in:

setLPFee()

setMarketingFee()

setDivideFee()

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

RECOMMENDATION

Provide information strings for **require** related errors.



Identifier	Definition	Severity
COD-06	Unknown externally owned account	Medium ●

An externally owned account (EOA) has no code, and one can send messages from an externally owned account by creating and signing a transaction.

0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c

0xDf0fa37b0d9992eD34F373607C14BdE5Dfe52C75

0x55d398326f99059fF775485246999027B3197955

INTERFI 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.



Identifier	Definition	Severity
COD-09	Missing contract balance withdraw	

Smart contract may collect tokens, and ethers from external addresses. Some swap, and liquidity-add events may accumulate residual ethers, and tokens.

RECOMMENDATION

Add `withdraw()` function to take out tokens and ethers from the contract.



Identifier	Definition	Severity
COD-10	Third Party Dependencies	Minor 

Smart contract is interacting with third party protocols e.g., Pancakeswap DEX, Iterable Mapping, Open Zeppelin Libraries. The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised, and exploited. Moreover, upgrades in third parties can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

RECOMMENDATION

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




Identifier	Definition	Severity
VOL-01	Irrelevant code	Unknown 🟤

Redundant code in SafeMath.

RECOMMENDATION

Remove redundant and dead code.



Identifier	Definition	Severity
VOL-02	Typographical Error	Minor 

Did you mean setFundAddress?

setFunAddress

Did you mean setDividendFee?

setDivideFee


INTERFI
CONFIDENTIAL

INTERFI
CONFIDENTIAL

RECOMMENDATION

Fix typographical errors.



Identifier	Definition	Severity
COM-04	Checks Effects Interactions	Minor 

Review Checks Effects Interactions for below mentioned functions:

`process()`

RECOMMENDATION

Set caller requirements, and/or define require statements to restrict access.



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