



SMART CONTRACT AUDIT



interfinetwork



hello@interfi.network



<https://interfi.network>

PREPARED FOR

CONVEYOR ROUTER V1

(CONVEYOR LABS)



INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	Conveyor Labs
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	
Blockchain	Ethereum Chain
Centralization	Active ownership
Commit	α40409e9abc31c035a1115e6093322c8b994098d
Website	https://app.conveyor.finance
	https://conveyor.finance
Twitter	https://twitter.com/conveyorlabs
Discord	https://discord.gg/w4hNNzAax
Report Date	September 09, 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 ●	Major ●	Medium ●	Minor ●	Unknown ●
Open	0	0	0	0	0
Acknowledged	1	0	1	3	2
Resolved	0	0	1	2	0
Critical ● Privileges	Withdraw, Upgrade Multi-call, Initialize Affiliate				

 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.

 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 GRAPH12

MANUAL REVIEW13

DISCLAIMERS 26

ABOUT INTERFI NETWORK 29


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 Conveyor Labs 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:

- ConveyorRouterV1.sol

 Contract dependencies on ConveyorMath and ConveyorSwapCallbacks are not audited due to being out-of-scope. 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	
Not available	
Contract Name	ConveyorRouterV1
Compiler Version	0.8.19
License	BUSL-1.1



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

- 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

```

| **IConveyorRouterV1** | Interface | |||
| L | swapExactTokenForToken | External ! |  |NO ! |
| L | swapExactEthForToken | External ! |  |NO ! |
| L | swapExactTokenForEth | External ! |  |NO ! |
| L | initializeAffiliate | External ! |  |NO ! |
| L | initializeReferrer | External ! |  |NO ! |
| L | upgradeMulticall | External ! |  |NO ! |
| L | quoteSwapExactTokenForToken | External ! |  |NO ! |
| L | quoteSwapExactTokenForEth | External ! |  |NO ! |
| L | quoteSwapExactEthForToken | External ! |  |NO ! |
| L | withdraw | External ! |  |NO ! |
| L | CONVEYOR_MULTICALL | External ! | |NO ! |
| L | affiliates | External ! | |NO ! |
| L | referrers | External ! | |NO ! |
|||||
| **IConveyorMulticall** | Interface | |||
| L | executeMulticall | External ! |  |NO ! |
|||||

```



| ****ConveyorRouterV1**** | Implementation | IConveyorRouterV1 | ||

| ^L | <Constructor> | Public ! | 🚫 | NO ! |

| ^L | swapExactTokenForToken | Public ! | 🚫 | NO ! |

| ^L | swapExactEthForToken | Public ! | 🚫 | NO ! |

| ^L | swapExactTokenForEth | Public ! | 🚫 | NO ! |

| ^L | quoteSwapExactTokenForToken | External ! | 🚫 | NO ! |

| ^L | quoteSwapExactEthForToken | External ! | 🚫 | NO ! |

| ^L | quoteSwapExactTokenForEth | External ! | 🚫 | NO ! |

| ^L | _safeTransferETH | Internal 🔒 | 🔴 | |

| ^L | _withdrawEth | Internal 🔒 | 🔴 | |

| ^L | _depositEth | Internal 🔒 | 🔴 | |

| ^L | withdraw | External ! | 🔴 | onlyOwner |

| ^L | confirmTransferOwnership | External ! | 🔴 | NO ! |

| ^L | transferOwnership | External ! | 🔴 | onlyOwner |

| ^L | upgradeMulticall | External ! | 🚫 | onlyOwner |

| ^L | initializeAffiliate | External ! | 🔴 | onlyOwner |

| ^L | initializeReferrer | External ! | 🚫 | NO ! |

| ^L | <Receive Ether> | External ! | 🚫 | NO ! |

|||||

| ****ConveyorMulticall**** | Implementation | IConveyorMulticall, ConveyorSwapCallbacks | ||

| ^L | <Constructor> | Public ! | 🔴 | NO ! |

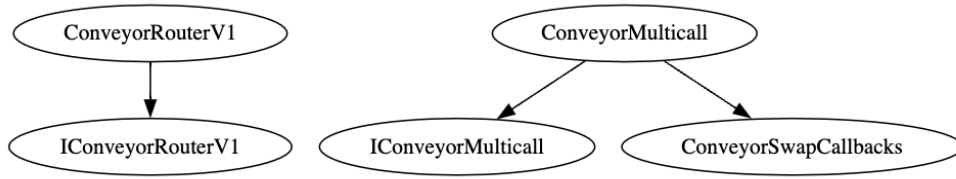
| ^L | executeMulticall | External ! | 🔴 | NO ! |

TERFI
CONFIDENTIAL

INTERFI
CONFIDENTIAL



INHERITANCE GRAPH



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



MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized and authorized privileges	Critical ●
CLR-01	Privileged role can change CONVEYOR_MULTICALL with upgradeMulticall()	
CLR-02	Privileged role can withdraw contract balance with withdraw()	

onlyOwner centralized privilege is provided to functions listed below:

```
withdraw()
transferOwnership()
upgradeMulticall()
initializeAffiliate()
```

tempOwner controlled privilege is provided to function listed below:

```
confirmTransferOwnership()
```

RECOMMENDATION

Smart contract owner has important privileges, including changing conveyor multicall address, and withdrawing all ETH from contract. While ownership transfer may be considered a feature, not limiting owner's power can have a single point of failure that compromises the security of the project. Deployer (original owner), contract owner, administrators, and all other privileged roles' private-keys/access-keys/admin-keys should be secured carefully. Ownership of the contract must be held by a trusted entity at any given point in time.



CEN-01 ACKNOWLEDGEMENT

Conveyor Labs team has commented that – project will implement multi-signature approach wherever possible to secure private-keys and admin-keys.

CLR-01 ACKNOWLEDGEMENT

Conveyor Labs team has commented that – reason for the `upgradeMulticall()` function is to future proof the compatibility of the protocol to new DEX architectures with new hooks required without redeploying the `ConveyorRouterV1`.

CLR-02 ACKNOWLEDGEMENT

Conveyor Labs team has commented that – only ETH stored in the contract are the protocol fees accumulated from swaps. Therefore, the protocol has privileged access to withdraw ETH from the contract. We acknowledge the security risk of protocol funds stolen if privileged keys are compromised.



Identifier	Definition	Severity
LOG-02	Note regarding front-running	Minor ●

Potential front-running happens when an attacker observes a transaction without setting restrictions on slippage or minimum output amount. The attacker can manipulate the operation by front-running a transaction to purchase assets and make profits by back-running a transaction to sell assets. Front-running is unavoidable on public blockchains.


RECOMMENDATION

Swap functions should be provided reasonable minimum output amounts, instead of zero.

ACKNOWLEDGEMENT

Conveyor Labs team has acknowledged this note.



Identifier	Definition	Severity
COD-01	Authorization through tx.origin	Minor 

Constructor uses tx.origin to set owner. This can be a security risk if the contract is created by another contract, as tx.origin refers to the original sender of the transaction and not the immediate caller.

RECOMMENDATION

Avoid authorizations via global variables wherever necessary.

RESOLUTION

According to Conveyor Labs team, ConveyorRouterV1 is deployed through a proxy contract, namely the Create3Factory. In order for them to deploy to deterministic addresses, they argued that setting the owner in the constructor to msg.sender is not a viable option. Options are to hardcode the address in the constructor, or to use tx.origin. They preferred using tx.origin over the former.



Identifier	Definition	
COD-03	Hardcoded values	

Constants like AFFILIATE_PERCENT and REFERRAL_PERCENT are hardcoded, making contract less flexible for future value changes.

```
uint128 internal constant AFFILIATE_PERCENT = 5534023222112865000;  
uint128 internal constant REFERRAL_PERCENT = 5534023222112865000;
```

RECOMMENDATION

If required by design, keep values changeable in future.

ACKNOWLEDGEMENT

Conveyor Labs team has commented that – these constants are chain agnostic, and intentionally hardcoded as the protocol will not be changing these values.



Identifier	Definition	Severity
COD-04	Re-entrancy	Medium 🟡

`executeMulticall()` uses low-level call to interact with an external contract. This action is technically prone re-entrancy attacks when `executeMulticall()` modifies contract state that the called target contract can influence.

RECOMMENDATION

Use mutex or re-entrancy guard to deter re-entrant calls. If possible, replace low-level calls like `call` with higher-level alternative transfer.

RESOLUTION

Conveyor Labs team has added `nonReentrant` modifier to `executeMulticall()`.



Identifier	Definition	
COD-08	Fallback function	

receive() function accepts incoming ETH without any restrictions. A malicious actor can flood the contract with small amounts of ETH to make contract's balance unpredictable. However, there's no direct gain with this attack vector, hence, it may not be considered a vulnerability.

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
COD-09	Assembly code risks	Unknown 🟤

Inline assembly is a way to access the Ethereum Virtual Machine (EVM) at low level. This bypasses several important safety features and checks of Solidity. Moreover, automated and manual checks are not confidently possible for inline assembly codes.

Below mentioned functions use inline assembly codes:

```
quoteSwapExactTokenForToken()
quoteSwapExactEthForToken()
quoteSwapExactTokenForEth()
_safeTransferETH()
_withdrawEth()
_depositEth()
upgradeMulticall()
executeMulticall()
```

"Inline assembly is risky. Assembly code is harder to read/audit, and it's also just harder to get right."

– Consensys (<https://consensys.io/diligence/blog/2019/07/return-data-length-validation-a-bug-we-missed/>)

RECOMMENDATION

Use high level Solidity constructs instead.

ACKNOWLEDGEMENT

Conveyor Labs team has commented that – Inline assembly is used in 2 scenarios in the code:

1. When doing basic operations where the risks can be minimized and gas savings maximized e.g., `transfer()`, `deposit()`, `withdraw()`, low level calls with return data size checks.
2. When required, e.g., in quoting functions to access `gas()` used by the transaction, as well as contract deployments to access the `create2` opcode.



Identifier	Definition	Severity
COD-10	Direct and indirect dependencies	Unknown 🟡

Smart contract is interacting with external and internal protocols e.g., Market Maker Callback contracts, External contracts, Web 3 applications, Open Zeppelin tools, Math libraries, Oracle library, etc. 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.

RECOMMENDATION

Inspect dependencies regularly, and mitigate severe impacts whenever necessary.

ACKNOWLEDGEMENT

Conveyor Labs team has commented that – libraries used in the ConveyorRouterV1 contract have all been audited. ConveyorMath has been audited by *Quantstamp* in a previous audit and has been unaltered. Oracle Libraries, Callbacks and Market Maker interactions are intentionally isolated into the ConveyorMulticall by design, isolating token approvals away from arbitrary calldata execution and unprotected callbacks. Conveyor Labs team will monitor contract interactions regularly, and plan amendments if necessary.



Identifier	Definition	Severity
COD-11	Inadequate access restrictions	Medium 🟡

Below mentioned function has no ownership or permission check, allowing for unauthorized changes:

```
initializeReferrer()
```

Function access must be restricted adequately to deter malicious actors, and unwanted state changes.

RECOMMENDATION

Conveyor Labs team has commented that - `initializeReferrer()`'s purpose is to allow EOA's to register as a referrer to accumulate 30% of fees on traders swaps used through their referral link. We acknowledge the potential for multiwallet spam filling up the referrers mapping in storage. However, we ensure that a single wallet cannot register as referral multiple times. Furthermore, `app.conveyor.finance` is heavily involved with syncing referrals to an external database and could eliminate the potential for a mass spam attack to generate any revenue on behalf of the attacker.



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

Smart contract uses function calls to update state, which can make it difficult to track and analyze changes to the contract over time.


RECOMMENDATION

Use events to track state changes. Events improve transparency and provide a more granular view of contract activity.

ACKNOWLEDGEMENT

Conveyor Labs team has commented that – this is an intentional design decision as the Routers fee revenue is solely dependent on the gas cost of using the router. We are able to monitor all activity off chain through different means such as gasless shadow events on forks of the chain. More on: <https://www.tryshadow.xyz>



Identifier	Definition	Severity
COM-01	Floating compiler status	Minor 

Compiler is set to ^0.8.19

INTERFI
CONFIDENTIAL

INTERFI
CONFIDENTIAL

RECOMMENDATION

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

RESOLUTION

Conveyor Labs team argued that the pragma is fixed to the minor version 0.8. Any breaking changes in solidity will bump the minor version.



Identifier	Definition	Severity
COM-04	Potential gas exhaustion	Minor ●

Mentioned functions are complex and do multiple state changes, which might incur high gas costs for users at times:

```
swapExactTokenForToken()
```

```
swapExactEthForToken()
```

```
swapExactTokenForEth()
```

RECOMMENDATION

Reduce function complexity to optimize gas usage in high gwei environment.

ACKNOWLEDGEMENT

Conveyor Labs team has acknowledged this finding, and commented that the router has fared pretty well against other routers in our internal tests. Here's the side-by-side comparison report:

Router	Gas	Tx	Block	Pool Type
Uniswap V2: Router 02 Swap	118,838	Tx hash	18111687	V2
ConveyorRouterV1	53,553	Foundry gas report	18111687	V2
Uniswap: Universal Router	142,547	Tx hash	18115370	V3
ConveyorRouterV1	56,893	Foundry gas report	18115370	V3



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

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