InterFi
NETWORK

SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

SENDR ESCROW CONTRACT

INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	Sendr
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	0xe36c5251153453f3c56b574698bbfd300baeda35
Blockchain	Ethereum
Centralization	Active Ownership
Commit	a910d0fc03daa99126ddc58f8216bf9b1b3338df
Website	
Report Date	October 20, 2024

ℹ 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	0	0	0	2	2
Resolved	1	1	1	2	0
Important Functions	`raiseDispute, voteOnDispute, createContract, signContract, releaseMilestone, disputeMilestone, disputeContract, voidContract`				
Noteworthy Privileges	`setSendrToken, setFee, setFeeWallet, setSendrTreasury, setVotingDuration, setVotingExtensionDuration, setThresholdPercent`				

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 Sendr 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:

○ SendrEscrow.sol

ℹ 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:

<table>
<tr><td colspan="2">Public Contract Link</td></tr>
<tr><td colspan="2">https://etherscan.io/address/0xe36c5251153453f3c56b574698bbfd300baeda35#code</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Contract Name</td><td>SendrEscrow</td></tr>
<tr><td>Compiler Version</td><td>0.8.20</td></tr>
<tr><td>License</td><td>MIT</td></tr>
</table>

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 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.

ℹ 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

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.

ℹ 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.

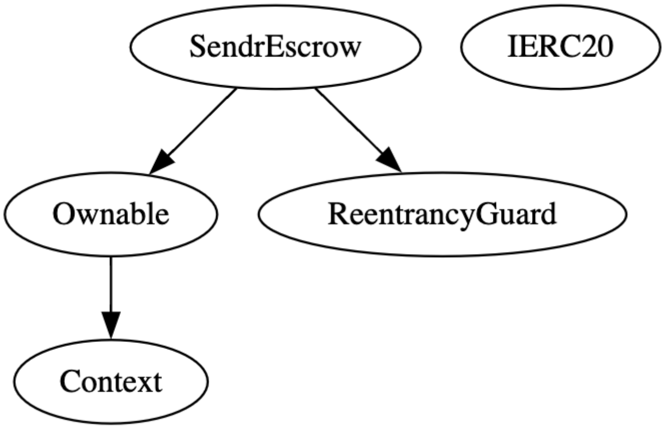
AUTOMATED ANALYSIS

Symbol	Definition
🛑	Function modifies state
💵	Function is payable
🔒	Function is internal
🔐	Function is private
❗	Function is important

```
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
| └ | _contextSuffixLength | Internal 🔒 |   | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | owner | Public ❗ |   |NO❗ |
| └ | _checkOwner | Internal 🔒 |   | |
| └ | renounceOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | _transferOwnership | Internal 🔒 | 🛑  | |
||||||
| **IERC20** | Interface |  |||
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |

| └ | transferFrom | External ❗ | 🛑  |NO❗ |

||||||

| **ReentrancyGuard** | Implementation |  |||

| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |

| └ | _nonReentrantBefore | Private 🔐 | 🛑  | |

| └ | _nonReentrantAfter | Private 🔐 | 🛑  | |

| └ | _reentrancyGuardEntered | Internal 🔒 |   | |

||||||

| **SendrEscrow** | Implementation | Ownable, ReentrancyGuard |||

| └ | <Constructor> | Public ❗ | 🛑  | Ownable |

| └ | setSendrToken | External ❗ | 🛑  | onlyOwner |

| └ | setFeeWallet | External ❗ | 🛑  | onlyOwner |

| └ | setFee | External ❗ | 🛑  | onlyOwner |

| └ | setSendrTreasury | External ❗ | 🛑  | onlyOwner |

| └ | setVotingDuration | External ❗ | 🛑  | onlyOwner |

| └ | setVotingExtensionDuration | External ❗ | 🛑  | onlyOwner |

| └ | setThresholdPercent | External ❗ | 🛑  | onlyOwner |

| └ | delegateVotes | External ❗ | 🛑  |NO❗ |

| └ | getPastVotes | Public ❗ |   |NO❗ |

| └ | raiseDispute | Public ❗ | 🛑  |NO❗ |

| └ | voteOnDispute | Public ❗ | 🛑  | nonReentrant |

| └ | _resolveDispute | Internal 🔒 | 🛑  | |

| └ | _sendFunds | Internal 🔒 | 🛑  | |

| └ | createContract | Public ❗ |  💵 | nonReentrant |

| └ | _generateMilestoneArray | Internal 🔒 |   | |

| └ | signContract | Public ❗ |  💵 | nonReentrant |

| └ | releaseMilestone | Public ❗ | 🛑  | nonReentrant |
```

└	disputeMilestone	Public ❗	🛑	NO❗
└	disputeContract	Public ❗	🛑	NO❗
└	voidContract	Public ❗	🛑	nonReentrant
└	getMilestones	External ❗		NO❗

INHERITANCE GRAPH



MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges of escrow contract	Major ●

Important `onlyOwner` centralized privileges are listed below:

```
renounceOwnership
transferOwnership
setSendrToken
setFeeWallet
setFee
setSendrTreasury
setVotingDuration
setVotingExtensionDuration
setThresholdPercent
```

RECOMMENDATION

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

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

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

RESOLUTION

Sendr team understands the importance of securing private keys and privileged access, as recommended. To reduce the risks of single-party control, they may use multi-signature wallets for executing sensitive contract functions, requiring multiple authorizations. This approach ensures greater security and accountability.

Identifier	Definition	Severity
LOG-02	Potential front-running	Minor 🟢

Front-running is possible when transactions can be predictably beneficial if ordered before other user's transactions. It is a concern primarily in public functions where the order of transactions can affect outcomes:

`voteOnDispute` Function: Since this function involves voting based on token balances, the potential for front-running exists where users may transfer tokens right before voting to affect the outcome.

All functions interacting with ERC-20 Tokens: Any function that allows interaction with ERC-20 tokens (`createContract` or `signContract`) can be susceptible to front-running, where users may attempt to time their transactions around price changes.

RECOMMENDATION

Functions that execute critical state changes should enforce minimum output thresholds. Setting these minimums above zero can deter malicious actors by reducing the predictability and profitability of front-running strategies.

Implement commit-reveal schemes or transaction ordering to protect against front-running.

ACKNOWLEDGEMENT

Front-running is not avoidable on public blockchains. Sendr team commented that, most EVM chains are prone to some sort of front-running and external manipulation.

Identifier	Definition	Severity
LOG-03	Re-entrancy	Critical 🔴
LOG-04	Checks-Effects-Interactions	

Below mentioned functions are used without Re-entrancy guard:

```
_sendFunds > _resolveDispute > voteOnDispute
releaseMilestone
createContract
signContract
voidContract
```

RECOMMENDATION

Use Checks-Effects-Interactions (CEI) pattern when transferring control to external entities. This design pattern ensures that all state changes are completed before external interactions occur. Additionally, implement re-entrancy guard to block recursive calls from external contracts.

RESOLUTION

Sendr team has added Re-entrancy Guard `nonReentrant` modifier to mentioned functions.

Identifier	Definition	Severity
LOG-05	Lack of function control checks	Medium 🟡

`releaseMilestone`: This function doesn't validate if both parties have agreed on the milestone release- Check if both parties have explicitly signed.

`voidContract`: This function doesn't validate if both parties have agreed before funds are returned- Check if both parties have explicitly agreed.

RECOMMENDATION

Implement stricter controls in functions to ensure that they cannot be executed unless all required conditions are met.

RESOLUTION

Smart contract checks if both parties have agreed to release milestone and void contract.

Identifier	Definition	Severity
COD-01	Potential denial of service (DoS)	Minor 🟢

Loops which iterate through arrays - can cause transactions to exceed block gas limit if there are too many milestones:

```
createContract
signContract
```

RECOMMENDATION

Implement gas-efficient patterns for functions that could potentially run into block gas limits, such as limiting the number of operations performed in a single transaction.

RESOLUTION

Sendr team has iterated that arrays are used as per logic requirement.

Identifier	Definition	Severity
COD-02	Timestamp dependence	Minor 🟢

Be aware that the timestamp of the block can be manipulated by miners. Since miners can slightly adjust the timestamp, they may influence contract outcomes to their advantage.

```
raiseDispute
voteOnDispute
signContract
```

RECOMMENDATION

Avoid relying solely on timestamp of the block for critical contract functions. Follow 15 seconds rule, and scale time dependent events accordingly.

ACKNOWLEDGEMENT

Sendr team commented that since the timestamp is not directly used to generate random numbers, the potential for miners to manipulate the outcome is minimal.

Identifier	Definition	Severity
COD-03	Note regarding governance attacks	Unknown ●

The voting mechanism can be susceptible to governance attacks if a few token holders possess a significant number of tokens. This can allow them to sway decisions in their favor consistently.

`delegateVotes` does not enforce any upper limits on delegation. If users delegate more than they actually hold later on, it may lead to incorrect voting powers at the time of voting.

ACKNOWLEDGEMENT

Sendr team commented that, smart contract utilizes a delegation system that can enhance how voting power is tracked and utilized, allowing for a more dynamic governance model.

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

Smart contract interacts with third-party protocols and external libraries, including OpenZeppelin's Ownable, ReentrancyGuard, ERC20, and governance utilities, as well as potential third-party ERC20 token contracts specified by users. The scope of this audit treats these dependencies as black boxes and assumes their functional correctness and security integrity. However, in practical scenarios, these external entities may be compromised or behave unpredictably due to bugs, malicious upgrades, or operational failures. Additionally, changes or upgrades in these dependencies, such as modifications to token mechanics or changes in the governance protocols, could significantly impact the contract's functionality, lead to increased transaction fees, or disrupt service continuity.

As such, continued diligence and monitoring of these dependencies are recommended to ensure ongoing contract security and performance.

RECOMMENDATION

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

ACKNOWLEDGEMENT

Sendr team will conduct regular inspections of third-party dependencies and promptly apply upgrades whenever necessary to ensure optimal performance and security.

Identifier	Definition	Severity
COM-01	Floating pragma	Minor 🟢

Compiler is set to `^0.8.0`

RECOMMENDATION

Pragma should be fixed to stable compiler version. Fixing pragma ensures compatibility and prevents the contract from being compiled with incompatible compiler versions.

RESOLUTION

Smart contract is deployed with stable compiler.

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.

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