InterFi NETWORK

SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

MARPTO

INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	Marpto
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	0x54d2473d282fe7ECEffCe0Ca0ACA0eB1a0cAFFfC
Blockchain	Binance Smart Chain
Centralization	Active ownership
Commit	7df1a44f54c67852d3b9d1759f75cbcdffcdc39b
Website	https://www.marpto.com/
Telegram	https://t.me/marptotoken/
X (Twitter)	https://x.com/marptotoken/
Report Date	February 21, 2024

INTERFI CONFIDENTIAL AUDIT REPORT

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	1	2	1	4	1
Acknowledged	0	1	0	1	0
Resolved	0	0	0	6	0
Major 🟠 Privileges	**Mint, Register Wh Router, Register LZ Router,** Set Send Version, Set Receive Version, Enable LayerZero, Enable Wormhole				

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTER
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDEN

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

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 Marpto 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:

- MRPTToken.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/0x54d2473d282fe7ECEffCe0Ca0ACA0eB1a0cAFFfC#code	
Contract Name	MRPTToken
Compiler Version	0.8.20
License	MIT

INTERFI AUDIT REPORT CONFIDENTIAL

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:

INTERFI AUDIT REPORT CONFIDENTIAL

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

INTERFI AUDIT REPORT CONFIDENTIAL

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.

INTERFI AUDIT REPORT CONFIDENTIAL

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.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTER
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDEN

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

INTERFI CONFIDENTIAL

```
| **MRPTToken** | Implementation | IMRPTToken, Ownable, ERC20, WormholeAdapter,
LayerZeroAdapter |||
| └ | <Constructor> | Public ❗ | 🛑  | Ownable ERC20 |
| └ | mint | External ❗ | 🛑  | onlyOwner |
| └ | transferFrom | External ❗ |  💵 |NO❗ |
| └ | transferFromWithCallback | External ❗ |  💵 |NO❗ |
| └ | circulatingSupply | External ❗ |   |NO❗ |
| └ | _normalizeAmount | Internal 🔒 |   | |
| └ | _deNormalizeAmount | Internal 🔒 |   | |
| └ | _transferFrom | Internal 🔒 | 🛑  | |
| └ | tryCallback | Public ❗ | 🛑  |NO❗ |
| └ | _remoteTransfer | Internal 🔒 | 🛑  | |
| └ | _remoteTransferWithCallback | Internal 🔒 | 🛑  | |
| └ | _receiveTransfer | Internal 🔒 | 🛑  | |
| └ | _receiveTransferWithCallback | Internal 🔒 | 🛑  | |
| └ | _isContract | Internal 🔒 |   | |
| └ | _nonblockingLzReceive | Internal 🔒 | 🛑  | |
| └ | _receiveLzTransferWithCallback | Internal 🔒 | 🛑  | |
```

| └ | _receiveWhTransferWithCallback | Internal 🔒 | 🛑 | |

| └ | _wormholeReceive | Internal 🔒 | 🛑 | |

||||||

| **IMRPTToken** | Interface | IAdapterCallParamStructure |||

| └ | transferFrom | External ❗ | 💵 |NO❗ |

| └ | transferFromWithCallback | External ❗ | 💵 |NO❗ |

| └ | circulatingSupply | External ❗ | |NO❗ |

||||||

| **IReceiveTransferCallback** | Interface | |||

| └ | onReceiveTransfer | External ❗ | 🛑 |NO❗ |

||||||

| **AddressTypeCast** | Library | |||

| └ | addressToBytes32 | Internal 🔒 | | |

| └ | bytes32ToAddress | Internal 🔒 | | |

||||||

| **Message** | Library | |||

| └ | payloadId | Internal 🔒 | | |

| └ | remote | Internal 🔒 | | |

| └ | encodeTransfer | Internal 🔒 | | |

| └ | encodeTransferWithCallback | Internal 🔒 | | |

| └ | decodeTransfer | Internal 🔒 | | |

| └ | decodeTransferWithCallback | Internal 🔒 | | |

||||||

| **Ownable** | Implementation | Context |||

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

| └ | owner | Public ❗ | |NO❗ |

| └ | _checkOwner | Internal 🔒 | | |

| └ | renounceOwnership | Public ❗ | 🛑 | onlyOwner |

| └ | transferOwnership | Public ❗ | 🛑 | onlyOwner |

| └ | _transferOwnership | Internal 🔒 | 🛑 | |

||||||

| **ERC20** | Implementation | Context, IERC20, IERC20Metadata |||

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

| └ | name | Public ❗ | |NO❗ |

| └ | symbol | Public ❗ | |NO❗ |

| └ | decimals | Public ❗ | |NO❗ |

| └ | totalSupply | Public ❗ | |NO❗ |

| └ | balanceOf | Public ❗ | |NO❗ |

| └ | transfer | Public ❗ | 🛑 |NO❗ |

| └ | allowance | Public ❗ | |NO❗ |

| └ | approve | Public ❗ | 🛑 |NO❗ |

| └ | transferFrom | Public ❗ | 🛑 |NO❗ |

| └ | increaseAllowance | Public ❗ | 🛑 |NO❗ |

| └ | decreaseAllowance | Public ❗ | 🛑 |NO❗ |

| └ | _transfer | Internal 🔒 | 🛑 | |

| └ | _mint | Internal 🔒 | 🛑 | |

| └ | _burn | Internal 🔒 | 🛑 | |

| └ | _approve | Internal 🔒 | 🛑 | |

| └ | _spendAllowance | Internal 🔒 | 🛑 | |

| └ | _beforeTokenTransfer | Internal 🔒 | 🛑 | |

| └ | _afterTokenTransfer | Internal 🔒 | 🛑 | |

||||||

| **VestingWallet** | Implementation | Context |||

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

| └ | <Receive Ether> | External ❗ | 💵 |NO❗ |

| └ | beneficiary | Public ❗ | |NO❗ |

| └ | start | Public ❗ | |NO❗ |

| └ | duration | Public ❗ | |NO❗ |

| └ | released | Public ❗ | |NO❗ |

| └ | released | Public ❗ | |NO❗ |

| └ | releasable | Public ❗ | |NO❗ |

| └ | releasable | Public ❗ | |NO❗ |

| └ | release | Public ❗ | 🛑 |NO❗ |

| └ | release | Public ❗ | 🛑 |NO❗ |

| └ | vestedAmount | Public ❗ | |NO❗ |

| └ | vestedAmount | Public ❗ | |NO❗ |

| └ | _vestingSchedule | Internal 🔒 | | |

||||||

| **ExcessivelySafeCall** | Library | |||

| └ | excessivelySafeCall | Internal 🔒 | 🛑 | |

| └ | excessivelySafeStaticCall | Internal 🔒 | | |

| └ | swapSelector | Internal 🔒 | | |

||||||

| **LayerZeroAdapter** | Implementation | Ownable, ILayerZeroReceiver, CommonErrorsAndEvents, LayerZeroAdapterErrorsAndEvents, ILayerZeroUserApplicationConfig |||

| └ | lzReceive | External ❗ | 🛑 |NO❗ |

| └ | setConfig | External ❗ | 🛑 |NO❗ |

| └ | setSendVersion | External ❗ | 🛑 | onlyOwner |

| └ | setReceiveVersion | External ❗ | 🛑 | onlyOwner |

| └ | forceResumeReceive | External ❗ | 🛑 | onlyOwner |

| └ | registerLzRouter | External ❗ | 🛑 | onlyOwner |

| └ | setLzDestGas | External ❗ | 🛑 | onlyOwner |

| └ | enableLayerZero | External ❗ | 🛑 | onlyOwner |

| └ | nonblockingLzReceive | Public ❗ | 🛑 |NO❗ |

| └ | retryMessage | Public ❗ | 💵 |NO❗ |

| └ | _ensureTrustedLzRouter | Internal 🔒 | | |

| └ | _ensureEndpointCaller | Internal 🔒 | | |

| └ | _blockingLzReceive | Internal 🔒 | 🛑 | |

| └ | _storeFailedMessage | Internal 🔒 | 🛑 | |

| └ | _lzSend | Internal 🔒 | 🛑 | |

| └ | _lzAdapterParam | Internal 🔒 | | |

| └ | _lzAdapterParam | Internal 🔒 | | |

| └ | _nonblockingLzReceive | Internal 🔒 | 🛑 | |

||||||

| **WormholeAdapter** | Implementation | Ownable, IWormholeReceiver, CommonErrorsAndEvents, WormholeAdapterErrorsAndEvents |||

| └ | registerWhRouter | External ❗️ | 🛑 | onlyOwner |

| └ | receiveWormholeMessages | External ❗️ | 💵 |NO❗️ |

INTERFI CONFIDENTIAL

| └ | enableWormhole | External ❗️ | 🛑 | onlyOwner |

INTERFI CONFIDENTIAL

| └ | _whSend | Internal 🔒 | 🛑 | |

| └ | _wormholeReceive | Internal 🔒 | 🛑 | |

||||||

| **IAdapterCallParamStructure** | Interface | |||

||||||

| **BytesLib** | Library | |||

| └ | concat | Internal 🔒 | | |

| └ | concatStorage | Internal 🔒 | 🛑 | |

| └ | slice | Internal 🔒 | | |

| └ | toAddress | Internal 🔒 | | |

| └ | toUint8 | Internal 🔒 | | |

| └ | toUint16 | Internal 🔒 | | |

| └ | toUint32 | Internal 🔒 | | |

| └ | toUint64 | Internal 🔒 | | |

| └ | toUint96 | Internal 🔒 | | |

| └ | toUint128 | Internal 🔒 | | |

| └ | toUint256 | Internal 🔒 | | |

| └ | toBytes32 | Internal 🔒 | | |

| └ | equal | Internal 🔒 | | |

| └ | equalStorage | Internal 🔒 | | |

||||||

| **Context** | Implementation | |||

| └ | _msgSender | Internal 🔒 | | |

| └ | _msgData | Internal 🔒 | | |

||||||

| **IERC20** | Interface | |||

| └ | totalSupply | External ❗️ | |NO❗️ |

| └ | balanceOf | External ❗️ | |NO❗️ |

| └ | transfer | External ❗️ | 🛑 |NO❗️ |

| └ | allowance | External ❗️ | |NO❗️ |

| └ | approve | External ❗️ | 🛑 |NO❗️ |

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

||||||

| **IERC20Metadata** | Interface | IERC20 |||

| └ | name | External ❗️ | |NO❗️ |

| └ | symbol | External ❗️ | |NO❗️ |

| └ | decimals | External ❗️ | |NO❗️ |

||||||

| **SafeERC20** | Library | |||

| └ | safeTransfer | Internal 🔒 | 🛑 | |

| └ | safeTransferFrom | Internal 🔒 | 🛑 | |

| └ | safeApprove | Internal 🔒 | 🛑 | |

| └ | safeIncreaseAllowance | Internal 🔒 | 🛑 | |

| └ | safeDecreaseAllowance | Internal 🔒 | 🛑 | |

| └ | forceApprove | Internal 🔒 | 🛑 | |

| └ | safePermit | Internal 🔒 | 🛑 | |

| └ | _callOptionalReturn | Private 🔐 | 🛑 | |

| └ | _callOptionalReturnBool | Private 🔐 | 🛑 | |

||||||

| **Address** | Library | |||

| └ | isContract | Internal 🔒 | | |

| └ | sendValue | Internal 🔒 | 🛑 | |

| └ | functionCall | Internal 🔒 | 🛑 | |

| └ | functionCall | Internal 🔒 | 🛑 | |

| └ | functionCallWithValue | Internal 🔒 | 🛑 | |

| └ | functionCallWithValue | Internal 🔒 | 🛑 | |

| └ | functionStaticCall | Internal 🔒 | | |

| └ | functionStaticCall | Internal 🔒 | | |

| └ | functionDelegateCall | Internal 🔒 | 🛑 | |

| └ | functionDelegateCall | Internal 🔒 | 🛑 | |

| └ | verifyCallResultFromTarget | Internal 🔒 | | |

| └ | verifyCallResult | Internal 🔒 | | |

| └ | _revert | Private 🔐 | | |

||||||

| **ILayerZeroEndpoint** | Interface | ILayerZeroUserApplicationConfig |||

| └ | send | External ❗️ | 💵 |NO❗️ |

| └ | receivePayload | External ❗️ | 🛑 |NO❗️ |

| └ | getInboundNonce | External ❗️ | |NO❗️ |

| └ | getOutboundNonce | External ❗️ | |NO❗️ |

| └ | estimateFees | External ❗️ | |NO❗️ |

| └ | getChainId | External ❗️ | |NO❗️ |

| └ | retryPayload | External ❗️ | 🛑 |NO❗️ |

| └ | hasStoredPayload | External ❗️ | |NO❗️ |

| └ | getSendLibraryAddress | External ❗️ | |NO❗️ |

| └ | getReceiveLibraryAddress | External ❗️ | |NO❗️ |

| └ | isSendingPayload | External ❗️ | |NO❗️ |

| └ | isReceivingPayload | External ❗️ | |NO❗️ |

| └ | getConfig | External ❗️ | |NO❗️ |

| └ | getSendVersion | External ❗️ | |NO❗️ |

| └ | getReceiveVersion | External ❗️ | |NO❗️ |

||||||

| **ILayerZeroReceiver** | Interface | |||

| └ | lzReceive | External ❗️ | 🛑 |NO❗️ |

||||||

INTERFI CONFIDENTIAL

| **ILayerZeroUserApplicationConfig** | Interface | |||

| └ | setConfig | External ❗️ | 🛑 |NO❗️ |

| └ | setSendVersion | External ❗️ | 🛑 |NO❗️ |

| └ | setReceiveVersion | External ❗️ | 🛑 |NO❗️ |

| └ | forceResumeReceive | External ❗️ | 🛑 |NO❗️ |

||||||

| **LayerZeroAdapterErrorsAndEvents** | Interface | |||

||||||

| **CommonErrorsAndEvents** | Interface | |||

||||||

| **WormholeAdapterErrorsAndEvents** | Interface | |||

||||||

| **IWormhole** | Interface | |||

| └ | publishMessage | External ❗️ | 💵 |NO❗️ |

| └ | initialize | External ❗️ | 🛑 |NO❗️ |

```
| └ | parseAndVerifyVM | External ❗️ |   |NO❗️ |
| └ | verifyVM | External ❗️ |   |NO❗️ |
| └ | verifySignatures | External ❗️ |   |NO❗️ |
| └ | parseVM | External ❗️ |   |NO❗️ |
| └ | quorum | External ❗️ |   |NO❗️ |
| └ | getGuardianSet | External ❗️ |   |NO❗️ |
| └ | getCurrentGuardianSetIndex | External ❗️ |   |NO❗️ |
| └ | getGuardianSetExpiry | External ❗️ |   |NO❗️ |
| └ | governanceActionIsConsumed | External ❗️ |   |NO❗️ |
| └ | isInitialized | External ❗️ |   |NO❗️ |
| └ | chainId | External ❗️ |   |NO❗️ |
| └ | isFork | External ❗️ |   |NO❗️ |
| └ | governanceChainId | External ❗️ |   |NO❗️ |
| └ | governanceContract | External ❗️ |   |NO❗️ |
| └ | messageFee | External ❗️ |   |NO❗️ |
| └ | evmChainId | External ❗️ |   |NO❗️ |
| └ | nextSequence | External ❗️ |   |NO❗️ |
| └ | parseContractUpgrade | External ❗️ |   |NO❗️ |
| └ | parseGuardianSetUpgrade | External ❗️ |   |NO❗️ |
| └ | parseSetMessageFee | External ❗️ |   |NO❗️ |
| └ | parseTransferFees | External ❗️ |   |NO❗️ |
| └ | parseRecoverChainId | External ❗️ |   |NO❗️ |
| └ | submitContractUpgrade | External ❗️ | 🛑  |NO❗️ |
| └ | submitSetMessageFee | External ❗️ | 🛑  |NO❗️ |
| └ | submitNewGuardianSet | External ❗️ | 🛑  |NO❗️ |
| └ | submitTransferFees | External ❗️ | 🛑  |NO❗️ |
| └ | submitRecoverChainId | External ❗️ | 🛑  |NO❗️ |
```

INTERFI CONFIDENTIAL

||||||

| **IWormholeRelayerBase** | Interface | |||

| └ | getRegisteredWormholeRelayerContract | External ❗️ | |NO❗️ |

||||||

| **IWormholeRelayerSend** | Interface | IWormholeRelayerBase |||

| └ | sendPayloadToEvm | External ❗️ | 💵 |NO❗️ |

| └ | sendPayloadToEvm | External ❗️ | 💵 |NO❗️ |

| └ | sendVaasToEvm | External ❗️ | 💵 |NO❗️ |

| └ | sendVaasToEvm | External ❗️ | 💵 |NO❗️ |

| └ | sendToEvm | External ❗️ | 💵 |NO❗️ |

| └ | send | External ❗️ | 💵 |NO❗️ |

| └ | forwardPayloadToEvm | External ❗️ | 💵 |NO❗️ |

| └ | forwardVaasToEvm | External ❗️ | 💵 |NO❗️ |

| └ | forwardToEvm | External ❗️ | 💵 |NO❗️ |

| └ | forward | External ❗️ | 💵 |NO❗️ |

| └ | resendToEvm | External ❗️ | 💵 |NO❗️ |

| └ | resend | External ❗️ | 💵 |NO❗️ |

| └ | quoteEVMDeliveryPrice | External ❗️ | |NO❗️ |

| └ | quoteEVMDeliveryPrice | External ❗️ | |NO❗️ |

| └ | quoteDeliveryPrice | External ❗️ | |NO❗️ |

| └ | quoteNativeForChain | External ❗️ | |NO❗️ |

| └ | getDefaultDeliveryProvider | External ❗️ | |NO❗️ |

||||||

| **IWormholeRelayerDelivery** | Interface | IWormholeRelayerBase |||

| └ | deliver | External ❗️ | 💵 |NO❗️ |

||||||

| **IWormholeRelayer** | Interface | IWormholeRelayerDelivery, IWormholeRelayerSend |||

||||||

| **IWormholeReceiver** | Interface | |||

| └ | receiveWormholeMessages | External ❗️ | 💵 |NO❗️ |

||||||

| **IERC20Permit** | Interface | |||

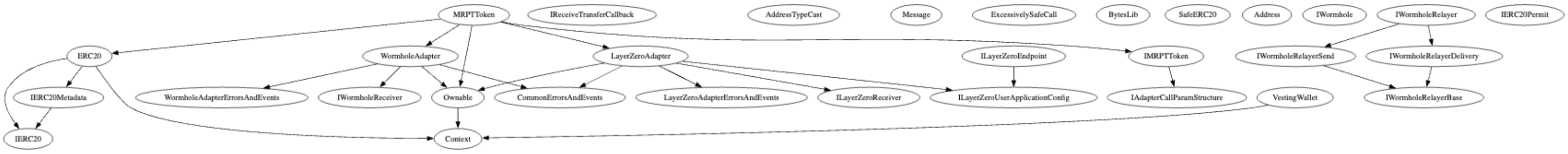
| └ | permit | External ❗️ | 🛑 |NO❗️ |

| └ | nonces | External ❗️ | |NO❗️ |

| └ | DOMAIN_SEPARATOR | External ❗️ | |NO❗️ |

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTER
NFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDEN

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 privileges	Major
CEN-09	Privileged role can mint tokens post-deployment	

Important `onlyOwner` centralized privileges are listed below:

```
mint()
renounceOwnership()
transferOwnership()
setSendVersion()
setReceiveVersion()
forceResumeReceive()
registerLzRouter()
setLzDestGas()
enableLayerZero()
registerWhRouter()
enableWormhole()
```

INTERFI CONFIDENTIAL

RECOMMENDATION

Deployers', owners', administrators', and all other privileged roles' private-keys/access-keys/admin-keys should be secured carefully. These entities can have a single point of failure that compromises the security of the project. Manage centralized and privileged roles carefully. It is recommended to:

Implement multi-signature wallets: Require multiple signatures from different parties to execute certain sensitive functions within contracts. This spreads control and reduces the risk of a single party having complete authority.

Use a decentralized governance model: Implement a governance model that enables token holders or other stakeholders to participate in decision-making processes. This can include voting on contract upgrades, parameter changes, or any other critical decisions that impact the contract's functioning.

ACKNOWLEDGEMENT

Marpto acknowledged to secure deployer and contract owners' private keys carefully. Marpto acknowledged to use multi-signature validation approach to manage centralization roles whenever possible.

INTERFI AUDIT REPORT CONFIDENTIAL

Identifier	Definition	Severity
CEN-02	Asset distribution	Minor

All of the minted assets are sent to vesting wallets and set addresses in `mint()` at owner's discretion. This can be an issue as the project owner can distribute tokens without consulting the community.

```
uint public mintable;
mintable = MAX_SUPPLY;

    function mint(address account, uint amount) external onlyOwner {
        mintable -= amount;
        _mint(account, amount);
    }
```

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTER
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.

RESOLUTION

Marpto project will distribute tokens after acquiring broader consensus, as per their pre-determined tokenomics. Marpto team commented that most of minted assets will be vested upon contract deployment.

Identifier	Definition	Severity
CEN-10	Inadequate access control	Critical

Mentioned functions should be provided adequate access control checks:

```
tryCallback()
release()
release()
lzReceive()
setConfig()
nonblockingLzReceive()
retryMessage()
lzReceive()
receiveWormholeMessages()
```

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

RECOMMENDATION

Provide adequate access control to stop unauthorized state changes. When contract state is changed with malicious intent, it introduces novel vulnerabilities, and hacks

Functions like `tryCallback()` can be set `internal` as well, when allowed by contract logic.

Identifier	Definition	Severity
MAR-01	Potential mint underflow	Minor

mint() function decreases the mintable amount without checking if smart contract has enough mintable supply left before minting new tokens. This may lead to underflows in the mintable variable, allowing minting of tokens beyond the intended MAX_SUPPLY.

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

RECOMMENDATION

Add require check ensures that minting cannot exceed the mintable supply.

```
function mint(address account, uint amount) external onlyOwner {
    require(mintable >= amount, "Not enough mintable supply");
    mintable -= amount;
    _mint(account, amount);
}
```

RESOLUTION

Marpto team argued that underflow protection is built into Solidity 0.8.0 and above. If this arithmetic operation will result in underflow, transaction will revert. However, it is still recommended to set explicit checks in mint() function.

Identifier	Definition	Severity
LOG-01	Validation of source data in LayerZero contracts	Major

LayerZero (LZ) contracts are part of the infrastructure for enabling cross-chain communication. Mentioned vulnerabilities are present in LZ contracts:

In `_nonblockingLzReceive` function, validate data's integrity and authenticity. Make sure message comes from a trusted source. When there's insufficient validation, it will lead to unauthorized actions being triggered on the receiving chain.

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

RECOMMENDATION

Validate source chain ID, source address, and payload. Use only trusted remote addresses or cryptographic proofs to verify authenticity.

Identifier	Definition	Severity
LOG-02	Potential front-running	Minor

Potential front-running 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 potentially vulnerable to front-running:

```
mint()
_startVesting()
```

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTER
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDEN

RECOMMENDATION

Use commit-reveal scheme to hide transactions until successful.

Identifier	Definition	Severity
LOG-03	Re-entrancy	Major

Below mentioned function is used without re-entrancy guard:

```
_startVesting()
creditTo()
_debitFrom()
release()
release()
```

Smart contract uses `ExcessivelySafeCall` for external calls, which is designed to mitigate re-entrancy risks by ensuring calls are made safely. This library method is used in `tryCallback` to make an external call to a callback function on another contract. While `ExcessivelySafeCall` is designed to be safe, re-entrancy vulnerability may occur in a non-traditional way, hence, it may be vulnerable to re-entrancy risks.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI
CONFIDENTIAL

RECOMMENDATION

Guard functions against re-entrancy attacks. Re-entrancy guard is used to prevent re-entrant calls. Learn more: https://consensys.github.io/smart-contract-best-practices/attacks/reentrancy/

NOTE

Marpto team argued that `_startVesting()` is callable in `constructor` only. Hence, re-entrancy control is not required.

Identifier	Definition	Severity
LOG-04	Price oracle manipulation	Minor

Functions depending on external price information, e.g., from DEXes or other sources may be vulnerable to price oracle manipulation.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTER
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDEN

RECOMMENDATION

Implementing a maximum percentage change between price updates can mitigate price manipulation risks.

Ensure that smart contract uses reliable and tamper-proof price feeds.

ACKNOWLEDGEMENT

Marpto team argued that no price data relies solely on on-chain price feeds, and kept the code as-is.

Identifier	Definition	Severity
COD-02	Timestamp manipulation and `block.timestamp` dependency	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. Ensure that use of timestamp logic can tolerate minor discrepancies.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTER
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDEN

RECOMMENDATION

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

RESOLUTION

Marpto project argued that smart contract is not using timestamp dependency to generate random numbers, or to compute chances. Miner manipulation should be minimal.

Identifier	Definition	Severity
COD-10	Direct and indirect dependencies	Unknown
COD-11	External contract interactions	
COD-12	Security of end-point contracts in LayerZero (LZ)	
COD-18	Security of `wormholeRelayer`	
COD-19	Reliance on LayerZero and Wormhole SDK	

Smart contract is interacting with third party protocols e.g., Market Makers, External Contracts, Web 3 Applications, *OpenZeppelin* 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.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTER
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDEN

Smart contract relies on external contracts `OFT`, `VestingWallet`, without explicit checks on these contracts' integrity or safety. Vulnerabilities will arise in Marpto smart contract when external contracts are hackable.

When using LayerZero infrastructure, vulnerabilities in the endpoint contracts will compromise the security of the entire cross-chain communication process.

RECOMMENDATION

Inspect all third-party dependencies and external contracts regularly, and mitigate severe impacts whenever necessary. Regularly audit and monitor LayerZero endpoint contracts for vulnerabilities. Only use established libraries and patterns.

Identifier	Definition	Severity
COD-13	Handling of message replay	Minor

Replay attacks involve an attacker re-sending a valid transaction to cause the intended action to be executed again, potentially leading to issues like double spending.

LayerZero code doesn't explicitly address replay protection.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTER
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDEN

RECOMMENDATION

Implement nonce checks or other mechanisms to ensure that each message can only be processed once. This can be handled by LayerZero infrastructure.

Identifier	Definition	Severity
COD-14	Potential signature replay attack / Message spoofing	Medium

`nonces` are is used for nonce management for accounts. Incorrect management or validation of nonces may lead to vulnerabilities like replay attacks. Ensure robust implementation and testing of nonce use.

Ensure message authenticity and integrity through adequate validation of messages received from LayerZero and Wormhole. Make sure received messages are not spoofed.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTER
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDEN

RECOMMENDATION

Both LayerZero and Wormhole use signatures to check the authenticity of messages. Implement following:

- Keep track of all processed nonces associated with a specific source address and chain ID.
- Before processing a message, check if the nonce has already been used. If it has, reject the message.
- Verify that received message adheres to the expected format, with the correct order and type of data fields.
- Check LayerZero documentation and review endpoint contracts to verify message authenticity.
- Use Wormhole SDK to decode and verify VAA (Verified Action Approvals).

Identifier	Definition	Severity
COD-15	Lack of event-driven architecture	Minor

Smart contract uses events in most functions, which is useful to track and analyze changes to the contract over time. However, not all functions emit events.

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

RECOMMENDATION

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

Identifier	Definition	Severity
COD-16	Note regarding `keccak256` secure hashing	Minor

Note that the `keccak256` function is not collision-resistant, and therefore there is a possibility of two different messages producing the same hash. Generating strong random input data, and properly securing and managing keys is recommended for fortification of `keccak256`.

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

Identifier	Definition
COD-17	Note regarding flash loan vulnerabilities

Smart contracts are not directly susceptible to flash loan attacks, which usually exploit some form of arbitrage opportunity. However, when smart contracts interact with malicious contracts, technically flash loan vulnerabilities can be introduced. For example, when "approved" underlying token contract turns out to be a malicious, it can be used to introduce flash-loan vulnerabilities. Be cautious while interacting with third-party contracts, tokens, and protocols.

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
VOL-01	Use of `delegatecall`	Minor

`delegatecall` is present, and is not clearly used in the smart contract.

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

RECOMMENDATION

Verify the user input and do not allow contract to perform `delegatecall` calls to untrusted contracts. Use of `delegatecall` in the contract is not recommended, as managing the storage layout in multiple contracts during logic update can be disruptive.

RESOLUTION

Marpto team has commented that – `delegatecall` has not been used in the smart contract. It is redundant.

Identifier	Definition	Severity
VOL-02	Assembly code	Minor

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.

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

RECOMMENDATION

Use high level Solidity constructs instead of `assembly`.

RESOLUTION

Marpto team has commented that – main assembly code is used for gas savings in byte manipulation and was written by *Consensys*, and is considered safe.

Identifier	Definition	Severity
COM-01	Multiple pragma directives	Minor
COM-02	Floating pragma	

Various compilers and floating pragma are used across all contracts.

INTERFI
CONFIDENTIAL

RECOMMENDATION

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

RESOLUTION

Marpto team has deployed the smart contract with stable compiler version. Multiple pragmas are still present in the smart contract.

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.

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

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.

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

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 INTER
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDEN

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

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTER
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDEN

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