InterFi
NETWORK

SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

MEMEWEWE

[TOKEN AND DISTRIBUTOR CONTRACTS]

INTERFI SMART CONTRACT AUDIT

INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	Memewewe
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Token	0x9B4F2fdd00B8340F15445Cc68C5191A81Ed5736F
Distributor	0xD951De12650bDC4d14f8E4A41938B4EEF29679c3
Blockchain	Base
Centralization	Active Ownership
Commit	c56916d9e94dd986930df1512c3bfeeff7f6f77e
Website	https://memewewe.me
Telegram	https://t.me/upsidedownmemetoken
X (Twitter)	https://x.com/memeweweme
Report Date	September 22, 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	6	0
Acknowledged	0	1	0	2	1
Resolved	0	0	1	2	0
Important Functions	`distributeDividend, claimDividend, process`				
Noteworthy Privileges	`_authorizeUpgrade, harvestWETH, clearStuckToken, setDistributor, updateTaxes, setTradingActive, setSwapbackStatus, setDistributorGas, setRewardToken, processNewRewardPrep`				

ℹ Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.

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

TABLE OF CONTENTS

TABLE OF CONTENTS 4

SCOPE OF WORK 5

AUDIT METHODOLOGY 6

RISK CATEGORIES 8

CENTRALIZED PRIVILEGES 9

MANUAL REVIEW 10

DISCLAIMERS 24

ABOUT INTERFI NETWORK 27

SCOPE OF WORK

InterFi was consulted by Memewewe 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:

- MEME_Token.sol
- MEME_RWD_Distributor.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:

Public Contract Link	
https://basescan.org/address/0x9b4f2fdd00b8340f15445cc68c5191a81ed5736f#code	
Contract Name	MEME_Token
Compiler Version	0.8.26
License	MIT

Public Contract Link	
https://basescan.org/address/0xd951de12650bdc4d14f8e4a41938b4eef29679c3#code	
Contract Name	MEME_RWD_Distributor
Compiler Version	0.8.26
License	MIT

AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of InterFi's auditing process and methodology:

CONNECT

- The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.

AUDIT

- Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:
  - Remix IDE Developer Tool
  - Open Zeppelin Code Analyzer
  - SWC Vulnerabilities Registry
  - DEX Dependencies, e.g., Pancakeswap, Uniswap
- Simulations are performed to identify centralized exploits causing contract and/or trade locks.
- A manual line-by-line analysis is performed to identify contract issues and centralized privileges. We may inspect below mentioned common contract vulnerabilities, and centralized exploits:

Centralized Exploits	○ 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.

MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges	Major 🟠
CEN-01-01	Privileged role can update distributor contract	
CEN-01-02	Privileged role can remove stuck tokens and ETH from contracts	
CEN-01-03	Privileged role must enable trading to allow token transfer	

Token

Important `onlyOwner` centralized privileges are listed below:

```
transferOwnership
_authorizeUpgrade
harvestWETH
clearStuckToken
setDistributor
updateTaxes
setLPReceiver
setOpsReceiver
setRwdReceiver
setTradingActive
setFeeExempt
setFreezeExempt
setRewardExempt
setSwapbackStatus
setDistributorGas
```

Distributor

```
transferOwnership
_authorizeUpgrade
harvestWETH
clearStuckToken
setRewardToken
processNewRewardPrep
```

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.

ACKNOWLEDGEMENT

Meme team argued that centralized and controlled privileges are used as required. Smart contracts utilize centralized administrative privileges to manage key functionalities such as initializing contracts, updating fee structures, and managing owner roles. These privileges are critical to the flexible and secure operation of the Memewewe system.

Meme team will use multi-signature wallets to manage centralization.

Identifier	Definition	Severity
CEN-02	Initial token allocation in token contract	Minor ●

Token

Upon deployment, all initially minted tokens are transferred to the contract deployer. It could be an issue as the deployer can distribute tokens without consulting the community.

```
        uint256 totalSupply = 100_000_000_000 * (10 ** 18);
        super._update(address(0), defaultReceiver_, totalSupply);
```

RECOMMENDATION

Establish transparent tokenomics model that involves community input in the decision-making process regarding token allocation.

ACKNOWLEDGEMENT

Meme team has clarified that initial token allocation will adhere strictly to pre-determined tokenomics outlined in project documentation.

Identifier	Definition	Severity
CEN-04	Use of proxy and upgradeable contracts	Medium 🟡

Privileged role can authorize contract upgrade. Contract upgradeability allows privileged roles to change current contract implementation.

```
contract MEME_Token is ERC20Upgradeable, OwnableUpgradeable, UUPSUpgradeable, ReentrancyGuard
contract MEME_RWD_Distributor is OwnableUpgradeable, UUPSUpgradeable, ReentrancyGuard {
```

RECOMMENDATION

Test and validate current contract thoroughly before deployment. While proxy contracts are great for robust deployments while maintaining the upgradeable flexibility, proxy codes are prone to new security or logical issues that may compromise the project.

RESOLUTION

Meme team emphasized the importance of upgradeable logic to address bugs and update contracts as needed. To ensure the contract cannot be reinitialized after the initial setup, they have incorporated `_disableInitializers` function. This addition effectively prevents any reinitialization, safeguarding the contract's integrity from unintended modifications.

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 called without setting restrictions on slippage or minimum output:

```
swapExactTokensForETHSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
_process
_update
```

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. Meme team commented that, most EVM chains are prone to some sort of front-running and external manipulation.

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.

RECOMMENDATION

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

Identifier	Definition	Severity
COD-07	Conformance to solidity writing guide in token contract	Minor 🟢

Token

Reward tax is currently being directed to the reward wallet, and the LP tax is funneled into the LP wallet. However, intended design is for reward tax should be to distribute it directly to the users and for LP tax – it should be sent straight to the liquidity pair, ensuring more efficient and transparent transaction processing.

RECOMMENDATION

Follow appropriate logic design for Reward tax and LP tax.

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

Smart contracts are interacting with third party protocols e.g., DEX routers, external contracts such as token and reward contracts, web3 applications, *OpenZeppelin* upgradeable and ERC20 libraries. 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 third party dependencies regularly, and mitigate severe impacts whenever necessary.

ACKNOWLEDGEMENT

Meme team will inspect third party dependencies regularly, and push upgrades whenever required.

Identifier	Definition	Severity
COD-11	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`.

COMMENT

Meme team comments that the keccak256 collision has little effect on the functionality as it's related to signed data, except for the storage layout that can be solved by creating test files to check storage slot collisions. `keccak256` function is widely adapted in cryptography, and its use is relatively safe.

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

Smart contracts use function calls to update state, which can make it difficult to track and analyze changes to the contract over time. Some functions are missing event emits

RECOMMENDATION

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

Identifier	Definition	Severity
COM-02	Multiple pragma directives	Minor ●

Multiple pragmas are used in smart contracts.

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 contracts are deployed with stable compiler.

Identifier	Definition	Severity
COM-03	Hardcoded gas use to distribute rewards in token contract	Minor 🟢

Token

Gas amount is set:

```
    uint256 public distribGas = 1_000_000;
```

RECOMMENDATION

Stop `rwdDistributor.process()` call in `_update()`. Users should claim their rewards manually through the function `claimDividend()`.

Identifier	Definition	Severity
COM-04	Gas optimization in distributor contract	Minor 🟢

Distributor

`process` function, which distributes dividends, could run out of gas if there are many shareholders.

RECOMMENDATION

Optimize gas usage and use mechanisms like checkpointing or off-chain calculations to handle large numbers of transactions.

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.

RECOMMENDATION

Use high level Solidity constructs instead of `assembly`.

RESOLUTION

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

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