InterFi NETWORK

SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

THE MIRROR PROTOCOL

INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	The Mirror Protocol
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	Multiple contracts
Blockchain	Binance Smart Chain
Centralization	Active ownership
Commit	52519c31951af480b108971d3be4b4a7dbb72de8
Website	https://themirrorprotocol.com/
Report Date	May 02, 2023

ℹ️ Verify the authenticity of this report on our website: https://www.github.com/interfinetwork

DEPLOYED CONTRACT LINKS

Proxy	ETH-MP	0x441f4393f4915F67F46B02baB276d3C0b0af4839
	Distributor	0x81800f765fE4eEa6aad68aD64b53004a0e734984
Proxy	ADA-MP	0x77202a18d95a0E619E465A2e1D9df2FB795170ac
	Distributor	0xb4d54c92b395276a314F8995824F8a7F859B6521
Proxy	XRP - MP	0x03D4C4faBdE3192E7dec00EeC1F526E0898E2EB7
	Distributor	0x00173881e0743033eC376B168E9240BE63c826d4
Proxy	MATIC - MP	0x10456a21AD61cb759A638d999D015b2a332e1f46
	Distributor	0x71CF5a41102f0c8D736fc9059e7Dde623fD016f1
Proxy	LINK-MP	0xd86a282eef75f37923Bdd5300fB65b8625146720
	Distributor	0x9680cD9BBB4A77D3d72d4F57DDb749CAe970Bab4

Current Implementation	MP	0x605C1063E3DdDfD9cEA815c3e1Df1A7D1BF5753A
	Distributor	0x24f47953E0c935120F90BcDC62BBc938DfffF240

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	1	6	0
Acknowledged	0	1	1	3	1
Resolved	0	0	0	0	0
Noteworthy Privileges	Check PAGE 09 for important centralized privileges				

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 The Mirror Protocol 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:**

- BBTFDynamicMultiDistributor.sol
- BBTFDynamicMultiDistributorStakingSupport.sol
- MirrorProtocol.sol
- MirrorProtocolAdmin.sol

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.

ℹ️ 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

BBTFDynamicMultiDistributor.sol

| **BBTFDynamicMultiDistributor** | Implementation | IBBTFMultiDistributor, Initializable, OwnableUpgradeable, LockableFunction |||

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

| └ | __BBTFDynamicMultiDistributor_init | Internal 🔒 | 🛑 | onlyInitializing |

| └ | __BBTFDynamicMultiDistributor_init_unchained | Internal 🔒 | 🛑 | onlyInitializing |

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

| └ | deposit | Public ! | 💵 |NO! |

| └ | excludeFromRewards | External ! | 🛑 | onlyOwnerOrToken |

| └ | setShares | Public ! | 🛑 | onlyToken |

| └ | setShare | Public ! | 🛑 | onlyToken |

| └ | _setShare | Internal 🔒 | 🛑 | |

| └ | process | External ! | 🛑 |NO! |

| └ | _process | Private 🔐 | 🛑 | lockFunction |

| └ | _assignBNB | Internal 🔒 | 🛑 | |

| └ | _swapTokenForBNB | Internal 🔒 | 🛑 | lockTheSwap |

| └ | _processBNBSwap | Internal 🔒 | 🛑 | |

| └ | _distributeDividend | Internal 🔒 | 🛑 | |

| └ | getCumulativeDividends | Internal 🔒 | | |

| └ | addShareholder | Internal 🔒 | 🛑 | |

| └ | removeShareholder | Internal 🔒 | 🛑 | |

| └ | _updateExclusions | Private 🔐 | 🛑 | |

| └ | claimDividend | External ❗ | 🛑 | NO❗ |

| └ | _claimDividend | Internal 🔒 | 🛑 | lockFunction |

| └ | getUnpaidEarnings | Public ❗ | | NO❗ |

| └ | updateMinIterationGas | External ❗ | 🛑 | onlyOwner |

| └ | updateGasForProcessing | External ❗ | 🛑 | onlyOwnerOrToken |

| └ | updateDynamicIterationGasAccuracy | External ❗ | 🛑 | onlyOwner |

| └ | setParentToken | External ❗ | 🛑 | onlyOwner |

| └ | clearStuckTokens | External ❗ | 🛑 | onlyOwner |

| └ | setBNBThreshold | External ❗ | 🛑 | onlyOwner |

| └ | updateRewardToken | External ❗ | 🛑 | onlyOwner |

| └ | addRewardToken | Public ❗ | 🛑 | onlyOwner |

| └ | removeRewardToken | External ❗ | 🛑 | onlyOwner |

||||||

BBTFDynamicMultiDistributorStakingSupport,sol

| **BBTFDynamicMultiDistributorStakingSupport** | Implementation |

BBTFDynamicMultiDistributor |||

| └ | initialize | Public ❗ | 🛑 | initializer |

| └ | __BBTFDynamicMultiDistributorStakingSupport_init | Internal 🔒 | 🛑 | onlyInitializing

|

| └ | __BBTFDynamicMultiDistributorStakingSupport_init_unchained | Internal 🔒 | 🛑 |

onlyInitializing |

| └ | setShare | Public ❗ | 🛑 | onlyToken |

| └ | setShares | Public ❗ | 🛑 | onlyToken |

| └ | updateStakingContract | External ❗ | 🛑 | onlyOwner |

MirrorProtocol.sol

| **MirrorProtocol** | Implementation | Initializable, IERC20MetadataUpgradeable, LPSwapSupportUpgradeable |||

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

| └ | initialize | Public ❗ | 🛑 | initializer |

| └ | __MirrorProtocol_init | Internal 🔒 | 🛑 | onlyInitializing |

| └ | __MirrorProtocol_init_unchained | Internal 🔒 | 🛑 | onlyInitializing |

| └ | <Fallback> | External ❗ | 💵 | NO❗ |

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

| └ | decimals | Public ❗ | | NO❗ |

| └ | balanceOf | Public ❗ | | NO❗ |

| └ | _balanceOf | Internal 🔒 | | |

| └ | transfer | Public ❗ | 🛑 | NO❗ |

| └ | allowance | Public ❗ | | NO❗ |

| └ | approve | Public ❗ | 🛑 | NO❗ |

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

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

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

| └ | updateGasForProcessing | Public ❗ | 🛑 | onlyOwner |

| └ | _approve | Internal 🔒 | 🛑 | |

| └ | _transfer | Private 🔐 | 🛑 | |

| └ | pushSwap | External ❗ | 🛑 | NO❗ |

| └ | selectSwapEvent | Private 🔐 | 🛑 | lockTheSwap |

| └ | _transferStandard | Private 🔐 | 🛑 | |

| └ | toggleTrading | External ❗ | 🛑 | onlyOwner |

| └ | toggleLaunchSettings | External ❗ | 🛑 | onlyOwner |

| └ | updateLaunchSettings | External ❗ | 🛑 | onlyOwner |

| └ | updateDistributorContract | External ❗ | 🛑 | onlyOwner |

| └ | excludeFromRewards | External ❗ | 🛑 | onlyOwner |

| └ | setAdminContract | External ❗ | 🛑 | onlyOwner |

| └ | batchAirdrop | External ❗ | 🛑 | NO❗ |

| └ | _batchAirdrop | Private 🔐 | 🛑 | lockTheSwap |

| └ | clearStuckTokens | External ❗ | 🛑 | onlyOwner |

| └ | clearStuckBNB | External ❗ | 🛑 | onlyOwner |

MirrorProtocolAdmin.sol

| **MirrorProtocolAdmin** | Implementation | IMirrorProtocolAdmin, Initializable, OwnableUpgradeable, UUPSAccessControlUpgradeable |||

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

| └ | initialize | Public ❗ | 🛑 | initializer |

| └ | __MirrorProtocolAdmin_init | Internal 🔒 | 🛑 | onlyInitializing |

| └ | __MirrorProtocolAdmin_init_unchained | Internal 🔒 | 🛑 | onlyInitializing |

| └ | updateProcessExempt | External ❗ | 🛑 | onlyOwner |

| └ | updateBlacklist | External ❗ | 🛑 | onlyOwner |

| └ | updatePauseExempt | External ❗ | 🛑 | onlyOwner |

| └ | updateLaunchSettingExempt | External ❗ | 🛑 | onlyOwner |

| └ | updateBatchTransferEnabledAddresses | External ❗ | 🛑 | onlyOwner |

| └ | setLIO | External ❗ | 🛑 | onlyOwner |

| └ | setOperationsWallet | External ❗ | 🛑 | onlyOwner |

| └ | setStakingAddress | External ❗ | 🛑 | onlyOwner |

| └ | setStakingPoolAddress | External ❗ | 🛑 | onlyOwner |

| └ | setRewardTreasuryWallet | External ❗ | 🛑 | onlyOwner |

| └ | excludeFromFee | External ❗ | 🛑 | onlyOwner |

| └ | getAMMPairs | External ❗ | | NO❗ |

| └ | updateBuyFees | External ❗ | 🛑 | onlyOwner |

| └ | updateSellFees | External ❗ | 🛑 | onlyOwner |

| └ | updateTransferFees | External ❗ | 🛑 | onlyOwner |

| └ | _updateFees | Internal 🔒 | 🛑 | |

| └ | registerPairAddress | Public ❗ | 🛑 | onlyOwner |

| └ | getAMMPairLength | External ❗ | 🛑 | NO❗ |

| └ | getAMMPairAtIndex | External ❗ | 🛑 | NO❗ |

| └ | isAMMPair | External ❗ | 🛑 | NO❗ |

| └ | processTransfer | External ❗ | | NO❗ |

| └ | _calculateFees | Private 🔐 | | |

| └ | clearStuckTokens | External ❗ | 🛑 | onlyOwner |

| └ | clearStuckBNB | External ❗ | 🛑 | onlyOwner |

Supporting

| **IMirrorProtocolAdmin** | Interface | |||

| └ | canBatchTransfer | External ❗ | | NO❗ |

| └ | isPauseExempt | External ❗ | | NO❗ |

| └ | isLaunchSettingExempt | External ❗ | | NO❗ |

| └ | getAMMPairs | External ❗ | | NO❗ |

| └ | isAMMPair | External ❗ | 🛑 | NO❗ |

| └ | lio | External ❗ | | NO❗ |

| └ | operationsWallet | External ❗ | | NO❗ |

| └ | stakingAddress | External ❗ | | NO❗ |

| └ | stakingPoolAddress | External ❗ | | NO❗ |

| └ | rewardTreasury | External ❗ | | NO❗ |

| └ | processTransfer | External ❗ | | NO❗ |

||||||

| **IBBTFMultiDistributor** | Interface | |||

| └ | deposit | External ❗ | 💵 |NO❗ |

| └ | process | External ❗ | 🛑 |NO❗ |

| └ | excludeFromRewards | External ❗ | 🛑 |NO❗ |

| └ | setShares | External ❗ | 🛑 |NO❗ |

| └ | setShare | External ❗ | 🛑 |NO❗ |

| └ | updateGasForProcessing | External ❗ | 🛑 |NO❗ |

||||||

| **LPSwapSupportUpgradeable** | Implementation | OwnableUpgradeable |||

| └ | __LPSwapSupport_init | Internal 🔒 | 🛑 | onlyInitializing |

| └ | __LPSwapSupport_init_unchained | Internal 🔒 | 🛑 | onlyInitializing |

| └ | _approve | Internal 🔒 | 🛑 | |

| └ | _balanceOf | Internal 🔒 | | |

| └ | getSwapRouters | External ❗ | |NO❗ |

| └ | getSwapRouterAtIndex | External ❗ | |NO❗ |

| └ | getSwapRouterCount | External ❗ | |NO❗ |

| └ | updateRouters | Public ❗ | 🛑 | onlyOwner |

| └ | updateMainRouter | Public ❗ | 🛑 | onlyOwner |

| └ | updateLiquidityReceiver | External ❗ | 🛑 | onlyOwner |

| └ | setSwapAndLiquifyEnabled | Public ❗ | 🛑 | onlyOwner |

| └ | setBuybackAndLiquifyEnabled | Public ❗ | 🛑 | onlyOwner |

| └ | swapAndLiquify | Internal 🔒 | 🛑 | |

| └ | swapTokensForCurrency | Internal 🔒 | 🛑 | |

| └ | swapTokensForCurrencyUnchecked | Private 🔐 | 🛑 | |

| └ | swapTokensForCurrencyAdv | Internal 🔒 | 🛑 | |

| └ | _swapTokensForCurrencyAdv | Private 🔐 | 🛑 | |

| └ | addLiquidity | Private 🔐 | 🛑 | |

| └ | swapCurrencyForTokens | Internal 🔒 | 🛑 | |

| └ | swapCurrencyForTokensAdv | Internal 🔒 | 🛑 | |

| └ | swapCurrencyForTokensUnchecked | Internal 🔒 | 🛑 | | |

| └ | _swapCurrencyForTokensAdv | Private 🔐 | 🛑 | | |

| └ | buybackAndLiquify | Internal 🔒 | 🛑 | | |

| └ | `forceBuybackAndLiquify` | External ❗ | 🛑 | onlyOwner |

| └ | `updateTokenSwapRange` | External ❗ | 🛑 | onlyOwner |

| └ | `updateCurrencySwapRange` | External ❗ | 🛑 | onlyOwner |

||||||

| **LockableFunction** | Implementation | | | |

||||||

MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges	Major 🟠
CEN-03	Privileged role performing blacklist	
CEN-05	Privileged role allowing transfer before trading is set to true	

Important `onlyOwner` centralized privileges are listed below:

BBTFDynamicMultiDistributor

`excludeFromRewards`
`updateMinIterationGas`
`updateGasForProcessing`
`updateDynamicIterationGasAccuracy`
`setParentToken`
`clearStuckTokens`
`setBNBThreshold`
`updateRewardToken`
`addRewardToken`
`removeRewardToken`

BBTFDynamicMultiDistributorStakingSupport

`updateStakingContract`

MirrorProtocol

`updateGasForProcessing`
`toggleTrading`
`toggleLaunchSettings`
`updateLaunchSettings`
`updateDistributorContract`
`excludeFromRewards`
`setAdminContract`
`batchAirdrop`
`clearStuckTokens`
`clearStuckBNB`

MirrorProtocolAdmin
updateProcessExempt
updateBlacklist
updatePauseExempt
updateLaunchSettingExempt
updateBatchTransferEnabledAddresses
setLIO
setOperationsWallet
setStakingAddress
setStakingPoolAddress
setRewardTreasuryWallet
excludeFromFee
updateBuyFees
updateSellFees
updateTransferFees
registerPairAddress
clearStuckTokens
clearStuckBNB

LPSwapSupportUpgradeable
updateRouters
updateMainRouter
updateLiquidityReceiver
setSwapAndLiquifyEnabled
setBuybackAndLiquifyEnabled
forceBuybackAndLiquify
updateTokenSwapRange
updateCurrencySwapRange

RECOMMENDATION

Contract creator, contract owner, administrator and all privileged roles' private keys should be secured carefully. Please refer to PAGE-09 CENTRALIZED PRIVILEGES for a detailed understanding.

ACKNOWLEDGEMENT

Mirror Protocol team acknowledged to use Gnosis multi-sig protocol to manage centralized privileges.

Identifier	Definition	Severity
CEN-02	Initial asset distribution of Mirror Protocol token	Minor

All of the initially minted assets are sent to the project owner when deploying the contract. This can be an issue as the project owner can distribute tokens without consulting the community.

```
totalSupply = _totalSupply;
_balances[_tokenOwner] = totalSupply;
emit Transfer(address(0), _tokenOwner, totalSupply);
```


RECOMMENDATION

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

Identifier	Definition	Severity
CEN-04	Privileged role receiving LP tokens	Minor

Smart contract function `addLiqudity()` sends liquidity to `liquidityReceiver`.

```
function addLiquidity(uint256 tokenAmount, uint256 cAmount) private {
    uint256 routerIndex = (++_currentRouterIndex) % swapRouters.length();
    _currentRouterIndex = routerIndex;
    IUniswapV2Router02 swapRouter = IUniswapV2Router02(swapRouters.at(routerIndex));

    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(swapRouter), tokenAmount);

    // add the liquidity
    swapRouter.addLiquidityETH{value: cAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        liquidityReceiver,
        block.timestamp
    );
}
```


RECOMMENDATION

Send LP tokens to dead address or unreachable address.

Identifier	Definition	Severity
CEN-06	Privileged role modifying pair and router	Minor

Privileged role can call `registerPairAddress(), updateRouters(), and updateMainRouter()`

RECOMMENDATION

The current trading pair, e.g., Pancakeswap or Uniswap pair should not be removed from automated market makers.

Identifier	Definition	Severity
CEN-09	Use of proxy and upgradeable contracts	Medium 🟡

Privileged role can initiate smart contract implementation. Contract upgradeability allows privileged roles to change current contract implementation. Below mentioned codes can be re-initialized to push upgrades:

```
BBTFDynamicMultiDistributor.sol
BBTFDynamicMultiDistributorStakingSupport.sol
MirrorProtocol.sol
MirrorProtocolAdmin.sol
```

Mirror Protocol team has added `_disableInitializers` in upgradeable implementation to add a safety measure that prevents initializer functions from being called more than once, reducing the risk of unintended behavior or vulnerabilities.

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.

Identifier	Definition	Severity
LOG-01	Lack of appropriate arbitrary boundaries	Minor

Below mentioned functions are set without any arbitrary boundaries.

`updateMinIterationGas`
`updateGasForProcessing`
`setBNBThreshold`
`updateBuyFees`
`updateSellFees`
`updateTransferFees`

RECOMMENDATION

These functions should be provided appropriate upper and lower boundaries, e.g., fees should only be allowed to change within pre-set values.

Identifier	Definition	Severity
LOG-02	Potential front-running	Minor

Potential front-running also classified as – sandwich attack happens when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by front-running a transaction to purchase assets and make profits by back-running a transaction to sell assets. Below mentioned functions are called without setting restrictions on slippage or minimum output:

```
addLiquidityETH
swapExactETHForTokensSupportingFeeOnTransferTokens
swapExactTokensForETHSupportingFeeOnTransferTokens
```


RECOMMENDATION

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

Identifier	Definition	Severity
LOG-03	Re-entrancy and access control	Medium 🟡

Although contracts use a `lockTheSwap` modifier, it is not sufficient as it only sets a boolean flag, but does not prevent any re-entrancy attacks in a standard way.

In addition to the above comment, below mentioned functions are used without a re-entrancy guard or a proper access control:

```
deposit
claimDividend
pushSwap
```

`processTransfer` in `MirrorProtocolAdmin` is marked external for gas savings purpose. It can be marked `internal`.

Some functions in `LPSwapSupportUpgradeable` like `swapTokensForCurrencyUnchecked`, `addLiquidity`, and `_swapCurrencyForTokensAdv` are declared as private. If the contract is intended to be inherited and those functions are meant to be used or overridden in child contracts, consider changing their visibility to internal.

RECOMMENDATION

Use Checks Effects Interactions pattern when handing over the flow to an external entity and/or 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/

Identifier	Definition	Severity
COD-02	Timestamp manipulation via `block.timestamp`	Minor 🟢

Be aware that the timestamp of the block can be manipulated by a miner. When the contract uses the timestamp to seed a random number, the miner can actually post a timestamp within 15 seconds of the block being validated, effectively allowing the miner to precompute an option more favorable to their chances.

RECOMMENDATION

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

Identifier	Definition	Severity
COD-04	Missing or inaccurate error messages	

Some `require` statements in `BBTFDynamicMultiDistributor` are not provided error messages for users.

RECOMMENDATION

Provide accurate information strings for `require` related errors.

Identifier	Definition	Severity
COD-10	Third Party Dependencies	Unknown

Smart contract is interacting with third party protocols e.g., market makers, parent token, staking token, UUPS upgradeable contracts, front-end applications, Open Zeppelin tools. The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised, and exploited. Moreover, upgrades in third parties can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

RECOMMENDATION

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

ACKNOWLEDGEMENT

Mirror Protocol team will inspect these dependencies to minimize downtime from external intervention.

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.

Identifier	Definition	Severity
VOL-01	Irrelevant code	Minor

Redundant code is found, e.g.

In the `buybackAndLiquify`, there's no need to approve the transfer from the `buybackEscrowAddress` to the current contract since the current contract is the token itself. Instead, you can use the `_transfer` function.

RECOMMENDATION

Remove redundant code.

Identifier	Definition	Severity
COM-01	Floating compiler status	

Compilers are set to `^0.8.19`

RECOMMENDATION

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

Identifier	Definition	Severity
COM-04	Potential resource exhaustion errors	Minor

Below mentioned loops may throw out of gas errors upon executing:

`_rewardsTokens`
`rewardTokenAddresses`
`rewardTokenArray`
`ammPairs`
`arraySize in batchAirdrop`

RECOMMENDATION

Set upper bounds for multi-address calls.

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