SECURITY ASSESSMENT

MemeFi Token

December 3, 2023

Audit Status: Pass

Risk Analysis

Classifications of Manual Risk Results

Classification	Description
🟥 Critical	Danger or Potential Problems.
🟧 High	Be Careful or Fail test.
🟩 Low	Pass, Not-Detected or Safe Item.
ⓘ Informational	Function Detected

Manual Code Review Risk Results

Contract Security	Description
🟩 Buy Tax	4%
🟥 Sale Tax	4%
🟩 Cannot Buy	Pass
🟩 Cannot Sale	Pass
🟩 Max Tax	10%
ⓘ Modify Tax	Yes
🟩 Fee Check	Pass
🟩 Is Honeypot?	Not Detected
🟩 Trading Cooldown	Not Detected
🟥 Enable Trade?	Fail

Contract Security	Description
🟢 Pause Transfer?	Not Detected
🟢 Max Tx?	Pass
🟡 Is Anti Whale?	Detected
🟢 Is Anti Bot?	Not Detected
🟢 Is Blacklist?	Not Detected
🟢 Blacklist Check	Pass
🟢 is Whitelist?	No Detected
🟢 Can Mint?	Pass
🟢 Is Proxy?	Not Detected
🟢 Can Take Ownership?	Not Detected
🟢 Hidden Owner?	Not Detected
ℹ Owner	0xCBf4f550B4237f1a66ef91b513bE84f6220Ec24a
🟢 Self Destruct?	Not Detected
🟢 External Call?	Not Detected
🟢 Other?	Not Detected
🟢 Holders	4
🟡 Audit Confidence	Medium

The summary section reveals the strengths and weaknesses identified during the assessment, including any vulnerabilities or potential risks that may exist. It serves as a valuable snapshot of the overall security status of the audited project. However, it is highly recommended to read the entire security assessment report for a comprehensive understanding of the findings. The full report provides detailed insights into the assessment process, methodology, and specific recommendations for addressing the identified issues.

Project Overview

Token Summary

Parameter	Result
Address	0x4cedf0d7A40DE912a05659A5fDB8cED29fb2D2aa
Name	MemeFi
Token Tracker	MemeFi (MemeFiWillBeTheMotherOfAllMemesItsNotYourTypicalMe)
Decimals	18
Supply	100,000,000
Platform	Binance Smart Chain
compiler	v0.8.19+commit.7dd6d404
Contract Name	MASTER
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://bscscan.com/address/0x4cedf0d7A40DE912a05659A5fDB8cED29fb2D2aa#code
Payment Tx	Corporate

Main Contract Assessed
Contract Name

Name	Contract	Live
MemeFi	0x4cedf0d7A40DE912a05659A5fDB8cED29fb2D2aa	Yes

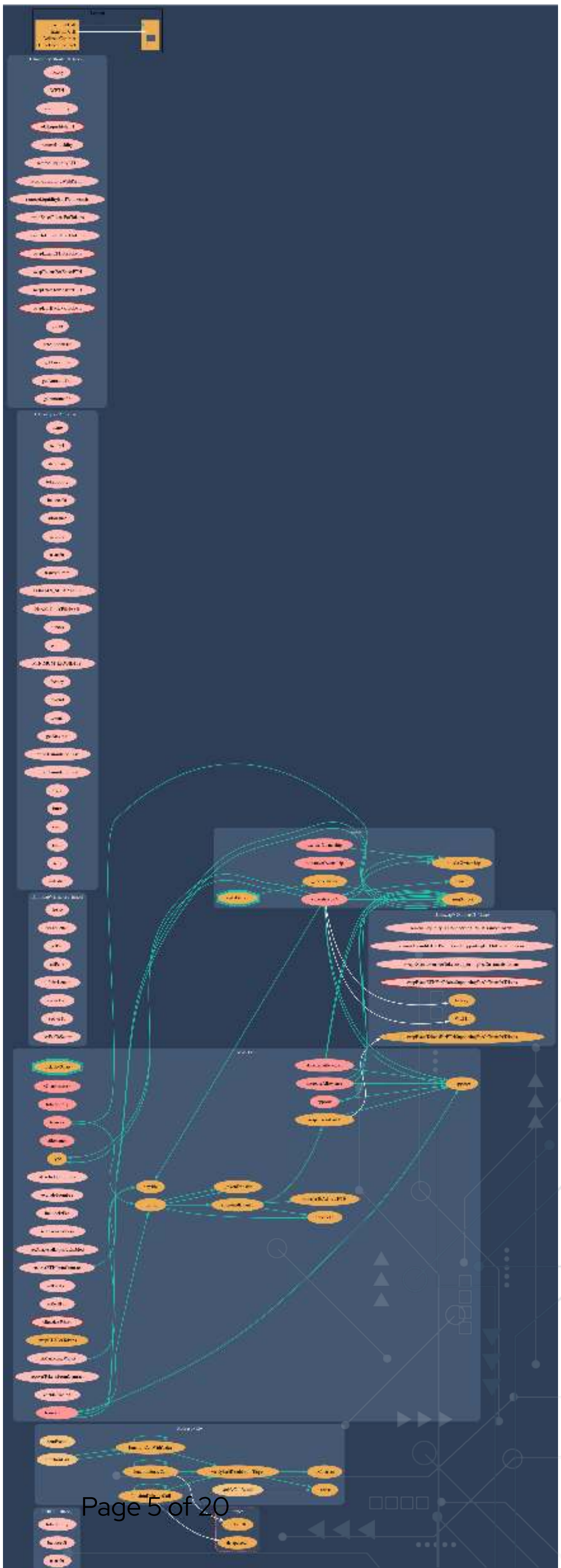
TestNet Contract was Not Assessed

Solidity Code Provided

SolID	File Sha-1	FileName
MEMEFI	5fd16556e5f94ed573c1ed7a4aeb47eb55afe7ca	master.sol

Call Graph

The Smart Contract Graph is a visual representation of the interconnectedness and relationships between smart contracts within a blockchain network. It provides a comprehensive view of the interactions and dependencies between different smart contracts, allowing developers and users to analyze and understand the flow of data and transactions within the network. The Smart Contract Graph enables better transparency, security, and efficiency in decentralized applications by facilitating the identification of potential vulnerabilities, optimizing contract execution, and enhancing overall network performance.



Smart Contract Vulnerability Checks

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) while overlaying a wide range of weakness variants that are specific to smart contracts.

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	master.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	master.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	master.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	master.sol	L: 7 C: 0
SWC-104	Pass	Unchecked Call Return Value.	master.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	master.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	master.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	master.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	master.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	master.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	master.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-111	Pass	Use of Deprecated Solidity Functions.	master.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	master.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	master.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	master.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	master.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	master.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	master.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	master.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	master.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randonmness.	master.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	master.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	master.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	master.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	master.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	master.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-126	Pass	Insufficient Gas Griefing.	master.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	master.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	master.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	master.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	master.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	master.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	master.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	master.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	master.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	master.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	master.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.

Smart Contract Vulnerability Details

SWC-103 - Floating Pragma.

CWE-664: Improper Control of a Resource Through its Lifetime.

References:

Description:

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation:

Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

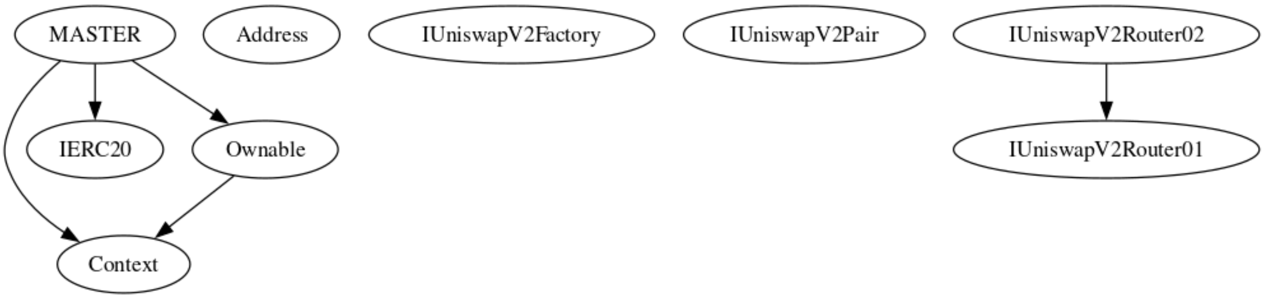
Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

References:

Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version.

Inheritance

Smart contract inheritance is a concept in blockchain programming where one smart contract can inherit properties and functionalities from another existing smart contract. This allows for code reuse and modularity, making the development process more efficient and scalable. Inheritance enables the child contract to access and utilize the variables, functions, and modifiers defined in the parent contract, thereby inheriting its behavior and characteristics. This feature is particularly useful in complex decentralized applications (dApps) where multiple contracts need to interact and share common functionalities. By leveraging smart contract inheritance, developers can create more organized and maintainable code structures, promoting code reusability and reducing redundancy.



MemeFiWillBeTheMotherOfAllMemesItsNotYourTypical Me-15 | Symbol Length Limitation due to Solidity Naming Standards.

Category	Severity	Location	Status
Logical Issue	● Medium	master.sol: L: 801 C: 14	Detected

Description

The Symbol is one of the most important part of the identity of a project, as industry standard this usually match the leng of stock market traditions. The Symbol used in contract is too long, this can create issues for most Dapps including uniswap, pancakeSwap and Metamask.
The current character limit for metamask is 11 and will be increased overtime to 20 characters, however APIS like uniswap,pancakeswap and coinmarketcap may have issues readins such symbols.

Recommendation

We advise removing the limiting the symbol to more industry standard naming to avoid issues with dapps and others. is suggested to use between 3 to 4 characters for a project, however for a coin is recommended no more than seven.

Mitigation

MemeFiWillBeTheMotherOfAllMemesItsNotYourTypicalMe

References:

Increase Token Symbol Length - Metamask

MemeFiWillBeTheMotherOfAllMemesItsNotYourTypicalMe-18 | Stop Transactions by using Enable Trade.

Category	Severity	Location	Status
Logical Issue	🔴 Critical	master.sol: L: 1169 C: 14	Detected

Description

Enable Trade is present on the following contract and when combined with Exclude from fees it can be considered a whitelist process, this will allow anyone to trade before others and can represent and issue for the holders.

Recommendation

We recommend the project owner to carefully review this function and avoid problems when performing both actions.

Mitigation

References:

Writing Clean Code for Solidity: Best Practices for Solidity Development

MemeFiWillBeTheMotherOfAllMemesItsNotYourTypicalMe-19 | Centralization Privileges of MemeFiWillBeTheMotherOfAllMemesItsNotYourTypicalMe

Category	Severity	Location	Status
Coding Style	🟡 Medium	master.sol: L: 0 C: 0	Detected

Description

In a smart contract, the concept of "onlyOwner" functions refers to certain functions that can only be executed by the owner or creator of the contract. These functions are typically designed to perform critical actions or modify sensitive data within the contract. By restricting access to these functions, the contract owner maintains control and ensures the integrity and security of the contract.

Function Name	Parameters	Visibility
renounceOwnership		Public
transferOwnership	address newOwner	Public
updateUniswapV2Router	address newAddress	public
enableTrading		external
recoverTokensFromContract		external
recoverETHfromContract		external
setSellFee		external
setBuyFeee		external
setMarketingWallet		external
setSwapAndLiquifyEnabled		external

Function Name	Parameters	Visibility
setTokensToSwap		external
includeInFee		external
excludeFromFee		public

Recommendation

Inheriting from Ownable and calling its constructor on yours ensures that the address deploying your contract is registered as the owner. The onlyOwner modifier makes a function revert if not called by the address registered as the owner. It is important that deployr or owner secure the credentials that has owner priviledge to ensure the security of the project.

Mitigation

References:

Guide to Ownership and Access Control in Solidity

Writing Clean Code for Solidity: Best Practices for Solidity Development

Technical Findings Summary

Classification of Risk

Severity	Description
Critical	Risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
High	Risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
Medium	Risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
Low	Risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the Project, but they may be less efficient than other solutions.
Informational	Errors are often recommended to improve the code's style or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
Critical	1	0	0
High	1	0	0
Medium	1	0	0
Low	0	0	0
Informational	0	0	0
Total	3	0	0

Social Media Checks

Social Media	URL	Result
Website	https://memefi.app/	Pass
Telegram	https://t.me/memefitokenofficial	Pass
Twitter	https://twitter.com/MemefiToken	Pass
Facebook	N/A	N/A
Reddit	N/A	N/A
Instagram	N/A	N/A
CoinGecko	N/A	N/A
Github	N/A	N/A
CMC	N/A	N/A
Other	N/A	N/A

From a security assessment standpoint, inspecting a project's social media presence is essential. It enables the evaluation of the project's reputation, credibility, and trustworthiness within the community. By analyzing the content shared, engagement levels, and the response to any security-related incidents, one can assess the project's commitment to security practices and its ability to handle potential threats.

Social Media Information Notes:

Auditor Notes: Website need improvements.

Project Owner Notes:

Assessment Results

Score Results

Review	Score
Overall Score	92/100
Auditor Score	80/100

Review by Section	Score
Manual Scan Score	36
SWC Scan Score	35
Advance Check Score	21

Our security assessment or audit score system for the smart contract and project follows a comprehensive evaluation process to ensure the highest level of security. The system assigns a score based on various security parameters and benchmarks, with a passing score set at 80 out of a total attainable score of 100.The assessment process includes a thorough review of the smart contracts codebase, architecture, and design principles. It examines potential vulnerabilities, such as code bugs, logical flaws, and potential attack vectors. The evaluation also considers the adherence to best practices and industry standards for secure coding. Additionally, the system assesses the projects overall security measures, including infrastructure security, data protection, and access controls. It evaluates the implementation of encryption, authentication mechanisms, and secure communication protocols. To achieve a passing score, the smart contract and project must attain a minimum of 80 points out of the total attainable score of 100. This ensures that the system has undergone a rigorous security assessment and meets the required standards for secure operation.

Audit Passed

Assessment Results

Important Notes:

- No issues or vulnerabilities were found.
- The contract has buy and sale tax as well as max wallet holding.
- Always DYOR, project code needs some improvements as described in the assessment.

Auditor Score =80
Audit Passed

AUDIT
PASSED

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that actagainst the nature of decentralization, such as explicit ownership or specialized access roles incombination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on howblock.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functionsbeing invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that mayresult in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to makethe codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code,such as a constructor assignment imposing different require statements on the input variables than a setterfunction.

Coding Best Practices

ERC 20 Conding Standards are a set of rules that each developer should follow to ensure the code meet a set of creterias and is readable by all the developers.

Disclaimer

The purpose of this disclaimer is to outline the responsibilities and limitations of the security assessment and smart contract audit conducted by Bladepool/CFG NINJA. By engaging our services, the project owner acknowledges and agrees to the following terms:

1. Limitation of Liability: Bladepool/CFG NINJA shall not be held liable for any damages, losses, or expenses incurred as a result of any contract malfunctions, vulnerabilities, or exploits discovered during the security assessment and smart contract audit. The project owner assumes full responsibility for any consequences arising from the use or implementation of the audited smart contract. 2. No Guarantee of Absolute Security: While Bladepool/CFG NINJA employs industry-standard practices and methodologies to identify potential security risks, it is important to note that no security assessment or smart contract audit can provide an absolute guarantee of security. The project owner acknowledges that there may still be unknown vulnerabilities or risks that are beyond the scope of our assessment. 3. Transfer of Responsibility: By engaging our services, the project owner agrees to assume full responsibility for addressing and mitigating any identified vulnerabilities or risks discovered during the security assessment and smart contract audit. It is the project owner s sole responsibility to ensure the proper implementation of necessary security measures and to address any identified issues promptly. 4. Compliance with Applicable Laws and Regulations: The project owner acknowledges and agrees to comply with all applicable laws, regulations, and industry standards related to the use and implementation of smart contracts. Bladepool/CFG NINJA shall not be held responsible for any non-compliance by the project owner. 5. Third-Party Services: The security assessment and smart contract audit conducted by Bladepool/CFG NINJA may involve the use of third-party tools, services, or technologies. While we exercise due diligence in selecting and utilizing these resources, we cannot be held liable for any issues or damages arising from the use of such third-party services. 6. Confidentiality: Bladepool/CFG NINJA maintains strict confidentiality regarding all information and data obtained during the security assessment and smart contract audit. However, we cannot guarantee the security of data transmitted over the internet or through any other means. 7. Not a Financial Advice: Bladepool/CFG NINJA please note that the information provided in the security assessment or audit should not be considered as financial advice. It is always recommended to consult with a financial professional or do thorough research before making any investment decisions.

By engaging our services, the project owner acknowledges and accepts these terms and releases Bladepool/CFG NINJA from any liability, claims, or damages arising from the security assessment and smart contract audit. It is recommended that the project owner consult legal counsel before entering into any agreement or contract.