AnalyticX

SMART CONTRACT AUDIT

@AnalyticX
@AnalyticX_Team
contact@analyticx.xyz
https://analyticx.xyz

PREPARED FOR:
MEMESTATION

SCAN TO VERIFY

ANALYTICX SMART CONTRACT AUDIT
AnalyticX

CONTENTS

INTRODUCTION

Project Name	MemeStation
Symbol	Memes
License Type	MIT
Blockchain	Ethereum
Contract	0xde0e9bd25dc63a629f40e1ec7959b57726e94d88
Decimals	18
Compiler Version	v0.8.19+commit.7dd6d404
Contract Link	https://etherscan.io/token/0xde0e9bd25dc63a629f40e1ec7959b57726e94d88#code
Website	https://www.memestationerc.com
Telegram	https://t.me/memestationerc
X (Twitter)	https://Twitter.com/Memestationeth
Report Date	September 18th, 2023

AUDIT METHODOLOGY

At AnalyticX, our smart contract audits are conducted with precision, following a set of rigorous standards and procedures. Collaboration with our clients is a cornerstone of our effective auditing process. Here's an overview of how we conduct our smart contract audits:

ONE

Our onboarding team initiates the process by gathering essential source codes and specifications. This initial step ensures a comprehensive understanding of the smart contract's size and scope, laying the foundation for a thorough audit.

TWO

AUDIT FOCUS

The primary objective of this audit was to ensure the security, resilience, and adherence to specifications of the Smart Contract System. The audit activities can be categorized into three key areas:

01 Security

Identification of security-related issues within each contract and the overall contract system.

02 Sound Architecture

- Evaluation of the system's architecture, aligning it with established smart contract best practices and general software best practices.

03 Code Correctness and Quality

A comprehensive review of the contract source code, with a focus on:

- Accuracy
- Readability
- Sections of code with high complexity
- Quantity and quality of test coverage

Each issue identified in this report has been assigned a severity level, categorized as follows:

ONE

Critical Severity Issues

These critical risks represent a significant threat, as they can be easily exploited, potentially leading to severe consequences such as asset loss, data manipulation, or both. It is imperative to address these issues promptly.

TWO

High Severity Issues

These risks, while not easily exploitable, are of critical importance to address. They bear an elevated risk of smart contract manipulation, which can escalate to a high-risk severity level.

THREE

Medium Severity Issues

Addressing these risks is essential as they come with an inherent potential for future exploits and hacks. While the impact on smart contract execution may vary, it's advisable to fix low-risk re-entrancy-related vulnerabilities to prevent possible exploits.

FOUR

Low Severity Issues

These risks, categorized as "Low," don't pose a significant threat to the contract or its users. They primarily involve code-style violations and deviations from standard practices. While not critical, it's advisable to highlight and address these issues for improved code quality.

AUDIT GOALS

At AnalyticX, our smart contract audits are conducted with precision, following a set of rigorous standards and procedures. Collaboration with our clients is a cornerstone of our effective auditing process. Here's an overview of how we conduct our smart contract audits:

01

Manual Audit

In this phase, the code underwent a thorough line-by-line examination by our developers. Additionally, we utilized Remix IDE's JavaScript VM and Kovan networks to test the contract's functionality.

02

Automated Audit

In our automated audit, we found no compiler warnings that could affect the contract. Our combined manual and automated assessment ensures the system's robustness, covering security, architecture, and code quality, with corresponding findings and recommendations in this report.

THREAT SUMMARY

It is engineered to aid users in detecting potential rug pull scams by offering a comprehensive examination of a smart contract's code and pinpointing any potential warning signs that could suggest a scam.

IS SOURCE CODE VERIFIED

The contract's source code is verified.
Source code verification provides transparency for users interacting with smart contracts. Block explorers validate the compiled code with the one on the blockchain. This also gives users a chance to audit the contracts.

PRESENCE OF MINTING FUNCTION

The contract cannot mint new tokens. The _mint function is not detected in the contract.
Mint functions are used to create new tokens and transfer them to the user's/owner's wallet to whom the tokens are minted. This increases the overall circulation of the tokens.

PRESENCE OF BURN FUNCTION

The tokens can not be burned in this contract.
Burn functions are used to increase the total value of the tokens by decreasing the total supply.

SOLIDITY PRAGMA VERSION

The contract can not be compiled with an older Solidity version.
Pragma versions decide the compiler version with which the contract can be compiled. Having older pragma versions means that the code may be compiled with outdated and vulnerable compiler versions, potentially introducing vulnerabilities and CVEs.

PROXY-BASED UPGRADABLE CONTRACT

This is not an upgradable contract.
Having upgradeable contracts or proxy patterns allows owners to make changes to the contract's functions, token circulation, and distribution.

OWNERS CANNOT BLACKLIST TOKENS OR USERS

Owners cannot blacklist tokens or users.
If the owner of a contract has permission to blacklist users or tokens, all the transactions related to those entities will be halted immediately.

IS ERC-20 TOKEN

The contract was found to be using ERC-20 token standard.
ERC-20 is the technical standard for fungible tokens that defines a set of properties that makes all the tokens similar in type and value.

THREAT SUMMARY

PAUSABLE CONTRACTS

This is not a Pausable contract.
If a contract is pausable, it allows privileged users or owners to halt the execution of certain critical functions of the contract in case malicious transactions are found.

CRITICAL ADMINISTRATIVE FUNCTIONS

Critical functions that add, update, or delete owner/admin addresses are not detected
These functions control the ownership of the contract and allow privileged users to add, update, or delete owner or administrative addresses. Owners are usually allowed to control all the critical aspects of the contract.

CONTRACT/TOKEN SELF DESTRUCT

The contract cannot be self-destructed by owners.
selfdestruct() is a special function in Solidity that destroys the contract and transfers all the remaining funds to the address specified during the call. This is usually access-control protected.

ERC20 RACE CONDITION

The contract is vulnerable to ERC-20 approve Race condition vulnerability.
ERC-20 approve function is vulnerable to a frontrunning attack which can be exploited by the token receiver to withdraw more tokens than the allowance. Proper mitigation steps should be implemented to prevent such vulnerabilities.

RENOUNCED OWNERSHIP

The contract's owner was not found.
Renounced ownership shows that the contract is truly decentralized and once deployed, it can't be manipulated by administrators.

USERS WITH TOKEN BALANCE MORE THAN 5%

Some addresses contains more than 5% of circulating token supply.
0x407993575c91ce7643a4d4ccacc9a98c36ee1bbe **20.0%**
Token distribution plays an important role when controlling the price of an asset.

OVERPOWERED OWNERS

The contracts are using 23 functions that can only be called by the owners.
Giving too many privileges to the owners via critical functions might put the user's funds at risk if the owners are compromised or if a rug-pulling attack happens.

THREAT SUMMARY

COOLDOWN FEATURE

The contract does not have a cooldown feature.
Cooldown functions are used to halt trading or other contract workflows for a certain amount of time so as to prevent users from repeatedly executing transactions or buying and selling tokens.

OWNERS WHITELISTING TOKENS/USERS

Owners can whitelist tokens or users.
If the owner of a contract has permission to whitelist users or tokens, it'll be unfair toward other users or the transaction flow may not be executed impartially.

OWNERS CAN SET/UPDATE FEES

Owners can not set or update Fees in the contract.

HARDCODED ADDRESSES

The contract was hardcoding addresses in the code. This may represent that those parameters can never be changed or updated unless it's a proxy contract. It is recommended to go through the code to know more about these hardcoded values and its use.

OWNERS UPDATING TOKEN BALANCE

The contract does not have any owner-controlled functions modifying token balances for users or the contract

OWNER WALLET TOKEN SUPPLY

The Owner's wallet contains 0 tokens which is less than 5% of the circulating token supply

FUNCTION RETRIEVING OWNERSHIP

No such functions were found
If this function exists, it is possible for the project owner to regain ownership even after relinquishing it.

AUDIT SUMMARY

AnalyticX has conducted comprehensive automated and manual analyses of Solidity Smart Contract code, meticulously reviewing it for common contract vulnerabilities and centralized Susceptibilities. Here's a brief summary of our audit findings:

Status	Critical	High	Medium	Low	Unknown
Open	0	0	0	0	0
Acknowledged	0	0	1	4	1
Resolved	0	0	0	0	0

In the realm of blockchain technology, it's essential to acknowledge that deployed smart contracts are not impervious to potential exploits, vulnerabilities, or hacks. These contracts operate within the context of emerging technologies, and as such, they inherently carry an ongoing level of risk. To gain a comprehensive understanding of the severity of these risks, vulnerabilities in the source code, and the limitations of our audit, we strongly urge you to thoroughly review the complete audit report.

One critical aspect to emphasize is that centralization privileges, regardless of their assigned risk status, exert a significant influence on the safety and security of smart contracts. These privileges can have a notable impact on the overall integrity of the contract.

Please keep in mind that blockchain technology and cryptographic assets are at the forefront of innovation, and the risks associated with them are dynamic. To delve into the specifics and access comprehensive information, we recommend referring to the disclaimer for full limitation of this audit.

CODE REVIEW

Transfer Fee 4%	Medium Risk
	Location #460-464
A fee has been discovered within the contract.	
We reduced risk level based on: **Owner: NullAddress**	Current Risk Low

Relevant Function Snippet

```
function transfer(address to, uint256 value) public virtual returns (bool) {
 address owner = _msgSender();
 _transfer(owner, to, value);
 return true;
}
```

Acknowledgement:
Contract is renounced, no one can change the transfer fee

Location in Code:
Transfer Fee: ERC20.transfer(address,uint256) (MemeStation.sol#460-464) - in nested function: _update
- in expression: fees = amount.mul(sellFeesTotal).div(100) - in expression: amount.mul(sellFeesTotal).div(100) - in expression: amount.mul(buyFeesTotal).div(100) - in expression: fees = amount.mul(buyFeesTotal).div(100)

AnalyticX

CODE REVIEW

Authorization through tx.origin	Medium Risk
	Location #1093-1216 and #1125-1129
This contract uses a type of authorization that may be exploited through other malicious contracts.	
Acknowledgement: Contract is renounced, can not be fixed but It's only used for limiting the purchase to one per block	

```
// MemeStation.sol - Function: _update(address, address, uint256)
// Lines: 1093-1216

function _update(address _address1, address _address2, uint256 _value) {
    // Authorization check using tx.origin
    require(
        _addressLastTransfer[tx.origin] < block.number,
        "_update:: Transfer Delay enabled. Only one purchase per block allowed."
    );

    // Rest of the function code goes here...

    // Lines: 1125-1129
}
```

CODE REVIEW

Blacklisting	High Risk
	Location #460-464 #505-510 #570-578
Wallets can be blacklisted from being able to transfer, swap or sell this token.	
We reduced risk level based on: **Owner: NullAddress**	Current Risk Low

Relevant Function Snippet

```
function transferFrom(address from, address to, uint256 value) public
virtual returns (bool) {
  address spender = _msgSender();
  _spendAllowance(from, spender, value);
  _transfer(from, to, value);
  return true;
}
function transfer(address to, uint256 value) public virtual returns (bool) {
  address owner = _msgSender();
  _transfer(owner, to, value);
  return true;
}
```

Acknowledgement:
Contract is renounced, backlisting is not possible anymore

Location in Code

Blacklisted function: ERC20.transfer(address,uint256)
MemeStation.sol#460-464)
- in expression: require(bool,string)(! blacklisted[from],Address is blacklisted)

Blacklisted function: ERC20.transferFrom(address,address,uint256)
(MemeStation.sol#505-510)
- in internal call: ERC20._transfer(address,address,uint256)
(MemeStation.sol#570-578)
- in expression require(bool,string)(! blacklisted[from],Address is blacklisted)

AnalyticX

CODE REVIEW

Transfer Limit	High Risk
	Location #460-464 #505-510
The max/min amount of token transferred can be limited (max could be set to 0).	
We reduced risk level based on: **Owner: NullAddress**	Current Risk Low

Relevant Function Snippet:

```
function transferFrom(address from, address to, uint256 value) public
virtual returns (bool) {
  address spender = _msgSender();
  _spendAllowance(from, spender, value);
  _transfer(from, to, value);
  return true;
}
function transfer(address to, uint256 value) public virtual returns (bool) {
  address owner = _msgSender();
  _transfer(owner, to, value);
  return true;
}
```

Acknowledgement:
Contract is renounced, Transfer limit can not be changed.

Location in Code
Transfer amount limits in:
ERC20.transferFrom(address,address,uint256) (MemeStation.sol#505-510)
- In expression: amount + balanceOf(to) <= maxHold
- In expression: amount + balanceOf(to)
- In expression: currentAllowance < value
- In expression: amount <= maxTx
- In expression: fromBalance < value
- In expression: amount + balanceOf(to)
- In expression: amount <= maxTx
- In expression: amount + balanceOf(to) <= maxHold

CODE REVIEW

Whitelisting	High Risk
	Location #460-464 #505-510
The contract have functionality of whitelisting	
We reduced risk level based on: **Owner: NullAddress**	Current Risk Low

Relevant Function Snippet

```
function transferFrom(address from, address to, uint256 value) public
virtual returns (bool) {
  address spender = _msgSender();
  _spendAllowance(from, spender, value);
  _transfer(from, to, value);
  return true;
}
function transfer(address to, uint256 value) public virtual returns (bool) {
  address owner = _msgSender();
  _transfer(owner, to, value);
  return true;
}
```

Acknowledgement:
Contract is renounced, Whitelisting is not possible now.

Location in Code

whitelisted function: ERC20.transfer(address,uint256) (MemeStation.sol#460-464)
whitelisted function: ERC20.transferFrom(address,address,uint256) (MemeStation.sol#505-510)
 - in internal call: ERC20._transfer(address,address,uint256) (MemeStation.sol#570-578)
 - in expression exemptFromFees[from]

AnalyticX

CODE REVIEW

Third Party Dependencies	Unknown Risk

The smart contract under audit interfaces with third-party protocols, such as Market Makers and OpenZeppelin tools. In the context of this audit, we treat these third-party entities as black boxes and assume their functional correctness.

1. **Assumption of Third-Party Functionality**: The audit treats third-party protocols as "black boxes," meaning they are assumed to work correctly and as intended. This is a common approach in smart contract audits because the audit's primary focus is on the code within the control of the audited project.
2. **Real-World Risks**: The report acknowledges that, in the real world, third-party protocols can be compromised or exploited by malicious actors. This poses a risk to the smart contract relying on these protocols.
3. **Impact of Upgrades**: Upgrades or changes in third-party protocols can have significant impacts on the functioning of the audited smart contract. This includes increased transaction fees, the deprecation of previous routers, and other potential disruptions.

Recommendation:
The report recommends that the audited project regularly inspect and monitor third-party dependencies. Regular checks are essential to proactively identify and mitigate any risks associated with these dependencies.

Acknowledgement:
The MemeStation team acknowledges the importance of this recommendation and commits to conducting regular checks on their third-party dependencies. This proactive approach to risk mitigation is a responsible step in maintaining the security and reliability of their smart contract.

DISCLAIMER

AnalyticX is committed to providing comprehensive audits of Solidity source codes, commonly known as smart contracts. In our meticulous analysis, we focus on identifying vulnerabilities, centralization exploits, and potential cybersecurity concerns within the framework and algorithms of the audited smart contract.

This limited report represents a condensed summary of our findings, reflecting industry best practices as of the report's date. It is important to emphasize that while we have undertaken rigorous efforts to assess and compile this report, it should not be solely relied upon. We cannot assume liability for the report's contents, production, or implications. It is imperative that you conduct your independent investigations before making any decisions related to the audited smart contract.

SCOPE OF AUDIT

Our audit scope predominantly revolves around the security analysis of the smart contract code itself. It does not extend to other areas beyond the programming language, such as the compiler layer. Furthermore, this audit report does not cover broader project aspects, including business models or legal compliance. It's crucial to note that this report does not guarantee the absolute absence of bugs within the smart contract.

CRYPTOGRAPHIC TOKENS AND TECHNICAL RISKS

Cryptographic tokens, particularly those associated with emerging technologies, inherently carry significant technical risks and uncertainties. By accessing our services, including reports and materials, you acknowledge that you do so at your own risk, accepting an "as-is, where-is, and as-available" basis. Please be aware that this audit report may include false positives, false negatives, and other unpredictable results.

CONFIDENTIALITY

The confidentiality of this report is paramount. It is subject to the terms and conditions outlined in the audit scope provided to the client. Without prior written consent from AnalyticX, this report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose.

DISCLAIMER

NO FINANCIAL ADVICE

This audit report should not be interpreted as an endorsement of any specific project or team, nor does it provide security guarantees. It is essential that third parties refrain from relying on this report for making financial decisions, whether related to the purchase or sale of products, services, or assets. The information presented in this report does not constitute investment advice, financial advice, trading advice, or any other form of advice. It is strongly advised not to use this audit report to inform investment or involvement decisions.

TECHNICAL DISCLAIMER

All our services, audit reports, smart contract audits, and other materials are provided "as is" and "as available." These come with all faults and defects, without any warranties, whether expressed, implied, statutory, or otherwise. AnalyticX disclaims all implied warranties, including merchantability, fitness for a particular purpose, title, and non-infringement, as well as any warranties arising from the course of dealing, usage, or trade practice. AnalyticX also disclaims liability and responsibility for any loss or damage resulting from this report, whether direct, indirect, special, punitive, or otherwise.

TIMELINESS OF CONTENT

It is essential to acknowledge that the content within this audit report may change without prior notice. AnalyticX does not guarantee or warrant the accuracy, timeliness, or completeness of any report accessed via the internet or other means.

LINKS TO OTHER WEBSITES

This audit report may provide access to websites and social accounts operated by entities other than AnalyticX. These hyperlinks are offered for reference and convenience. The content and operation of these websites and social accounts are solely the responsibility of the respective website and social account owners. AnalyticX disclaims any responsibility for the content or operation of such websites and social accounts, and shall have no liability for your use of third-party websites and social accounts linked from this report. You are solely responsible for determining the extent of your use of content on external websites and social accounts.

ABOUT ANALYTICX

AnalyticX is a leading provider of intelligent blockchain solutions, specializing in a wide range of services to enhance your blockchain project. Our expertise encompasses solidity development, comprehensive testing, and rigorous auditing services.

AnalyticX has a expertise across major public blockchains, including Ethereum, Binance Smart Chain, Cronos, Dogecoin, Polygon, Avalanche, Metis, Fantom, Bitcoin Cash, Velas, Oasis, and many more. We're committed to contributing to the growth and success of various blockchain ecosystems.

Our Team
Our dedicated team at AnalyticX is comprised of skilled engineers, developers, UI experts, and passionate blockchain enthusiasts. With a core team of 4 members and the support of over 6 casual contributors, we bring a wealth of experience and innovative thinking to every project we undertake.

We thank you for your continued support in our efforts to contribute to the Security of Blockchain and Crypto space. At AnalyticX, we're dedicated to helping you achieve your blockchain goals with intelligence and innovation.

GET IN TOUCH

- @AnalyticX_Team
- @AnalyticX (Community)
- contact@analyticx.xyz
- https://analyticx.xyz