



MemePad (MEPAD Token)

SMART CONTRACT AUDIT

07.05.2021

Made in Germany by Chainsulting.de



Table of contents

1. Disclaimer	3
2. About the Project and Company	4
2.1 Project Overview	5
3. Vulnerability & Risk Level.....	6
4. Auditing Strategy and Techniques Applied	7
4.1 Methodology.....	7
4.2 Used Code from other Frameworks/Smart Contracts	8
4.3 Tested Contract Files.....	8
4.4 Metrics / CallGraph	9
4.5 Metrics / Source Lines	10
4.6 Metrics / Capabilities.....	11
4.7 Metrics / Source Unites in Scope.....	12
5. Scope of Work.....	13
5.1 Manual and Automated Vulnerability Test	14
5.1.1 AntiWhale	15
5.2. SWC Attacks & Special Checks	16
6. Verify Claims	20
7. Executive Summary	25
8. Deployed Smart Contract.....	25



1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of MemePad Project. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (06.05.2021)	Layout
0.5 (07.05.2021)	Verify Claims and Test Deployment
0.6 (07.05.2021)	Testing SWC Checks
0.8 (07.05.2021)	Automated Security Testing Manual Security Testing
0.9 (07.05.2021)	Summary and Recommendation
1.0 (07.05.2021)	Final document

2. About the Project and Company

Company address:

Swim Company Ltd
Lavida Plus Officetel
Nguyen Van Linh Street
Tan Phong, District 7
Ho Chi Minh City
Vietnam

Website: <https://memepad.co>

Twitter: <https://twitter.com/MemePadLaunch>

Telegram: <https://t.co/bbjOTcRPQ?amp=1>



2.1 Project Overview

The first-ever IDO launchpad focusing exclusively on memecoins and other micro-cap coins. Memepad's platform allows whitelisted addresses to swap BNB or BUSD for an allocation in listed projects.

By launching on Memepad, you will get exposure to thousands of new potential buyers. Memepad is not only for memes, but also for coins created for charitable causes, as well as other micro-cap projects.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
 - ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
 - i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

4.2 Used Code from other Frameworks/Smart Contracts (direct imports)

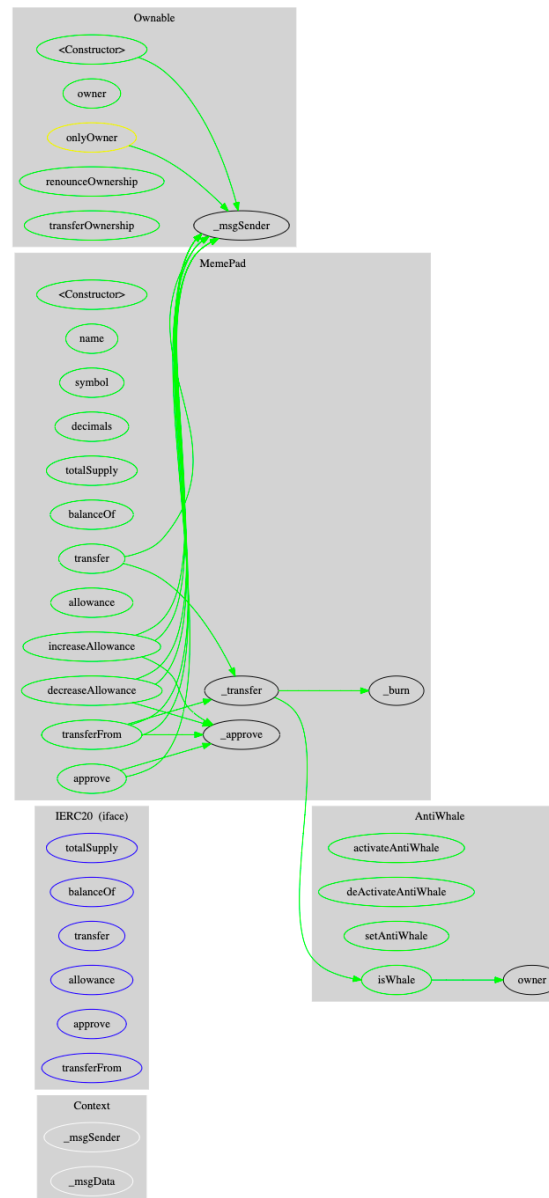
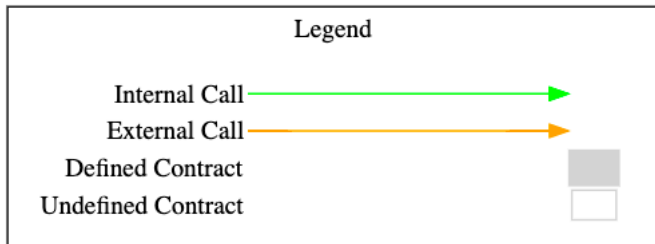
Dependency / Import Path	Source
@openzeppelin/contracts/token/ERC20/IERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/GSN/Context.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/GSN/Context.sol
@openzeppelin/contracts/token/ERC20/ERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol

4.3 Tested Contract Files

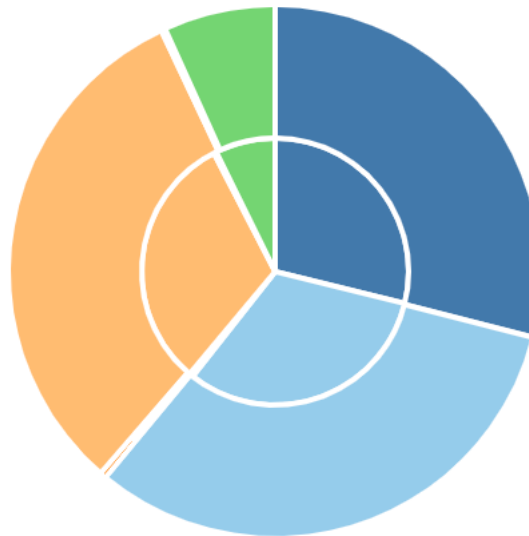
The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
MemePad.sol	3a4d9428cfc9972df2427bfb3422dfb3











4.4 Metrics / CallGraph



4.5 Metrics / Source Lines





4.6 Metrics / Capabilities


Solidity Versions observed		 Experimental Features		 Can Receive Funds		 Uses Assembly		 Has Destroyable Contracts			
<div>=0.8.4</div>				<div></div>		**** (0 asm blocks)		<div></div>			
 Transfers ETH		 Low-Level Calls		 DelegateCall		 Uses Hash Functions		 ECRrecover		 New/Create/Create2	
<div></div>		<div></div>		<div></div>		<div></div>		<div></div>		<div></div>	

Exposed Functions





This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

 Public	 Payable				
24	0				
External	Internal	Private	Pure	View	
6	23	0	0	13	

StateVariables

Total	 Public
12	4

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	MemePad.sol	4	1	538	436	186	255	133	
	Totals	4	1	538	436	186	255	133	

Legend: [☐]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

5. Scope of Work

The MemePad Team provided us with the file that needs to be tested. The scope of the audit is the MEPAD Token contract.

Following contracts with the direct imports has been tested:

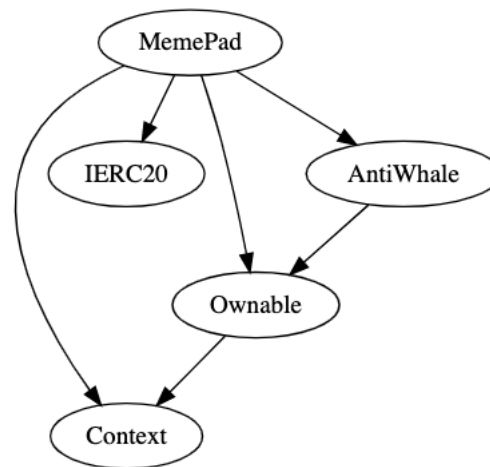
- MemePad.sol

The team put forward the following assumptions regarding the security, usage of the contracts:

Verify claims:

1. BEP-20 Token standard is correct implemented
2. Deployer cannot mint any new tokens.
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall smart contract security needs to be checked

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.



5.1 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, Chainsulting's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, Chainsulting's experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, Chainsulting's experts found **no Medium issues** in the code of the smart contract.

LOW ISSUES

5.1.1 AntiWhale

Severity: LOW

Status: Acknowledged

File(s) affected: MemePad.sol

Attack / Description	Code Snippet	Result/Recommendation
Owner is able to set the _limitWhale to 0 for a specific timeframe (_startDate - _endDate) and is not allowing transaction within the set timeframe.	<pre>Line 207 – 218 function isWhale(uint256 amount) public view returns (bool) { if (msg.sender == owner() antiWhaleActivated == false amount <= limitWhale) return false; if (block.timestamp >= startDate && block.timestamp <= endDate) return true; return false; }</pre>	<p>We demonstrated the possible _limitWhale 0 within claim 6.4 (Pause contract scenario). We consider the issue as low, as setting the _limitWhale to 0 would not be in the best interest of the team / project.</p> <p>If the team wants to disable AntiWhale or not use the function anymore, the only way to do it, would be the ownership transfer to address: 0x00dEaD</p> <p>and before the transfer to disable AntiWhale.</p>










INFORMATIONAL ISSUES

During the audit, Chainsulting's experts found **no Informational issues** in the code of the smart contract.



5.2. SWC Attacks & Special Checks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	


ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	


ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	X
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✓
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✓
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓

6. Verify Claims

6.1 BEP-20 Token standard is correct implemented

Status: tested and verified 

6.2 Deployer cannot mint any new tokens.

Status: tested and verified 

Max / Initial Supply: 400,000,000 (400M)


Code: Ln 254 - 255

```
uint256 constant maxCap = 400000000 * (10**18);
```

```
uint256 private _totalSupply = maxCap;
```

```
constructor() {  
    _balances[msg.sender] = maxCap; //At the moment of creation all tokens will go to the owner.  
}
```

6.3 Deployer cannot burn or lock user funds

Status: tested and verified 

Code: No burn or lock function

1. activateAntiWhale	→
2. approve	→
3. deActivateAntiWhale	→
4. decreaseAllowance	→
5. increaseAllowance	→
6. renounceOwnership	→
7. setAntiWhale	→
8. transfer	→
9. transferFrom	→
10. transferOwnership	→

6.4 Deployer cannot pause the contract

1. activateAntiWhale	→
2. approve	→
3. deActivateAntiWhale	→
4. decreaseAllowance	→
5. increaseAllowance	→
6. renounceOwnership	→
7. setAntiWhale	→
8. transfer	→
9. transferFrom	→
10. transferOwnership	→

6.4.1 Test Deployment of MEPAD Token

Contract: <https://ropsten.etherscan.io/address/0xbf7d39426f84ee43c5d62bacee1868b60b599213#code>

Tx : <https://ropsten.etherscan.io/tx/0xac85f7c2765a55f144753b7dbb83b7734e154711f4b61b34df399e187649b185>

Wallet A (1000 MEPAD Token)

<https://ropsten.etherscan.io/address/0xe1d80e7833f8b81d792922aecc0467ccfc3715bc>

6.4.2 Deployer is now activating AntiWhale with limit 0

Tx: <https://ropsten.etherscan.io/tx/0xac69a4a4497fdf628147065f957dcd2c4c0dd880ea14aa54da9125f0c6ff5ae3>

7. setAntiWhale

_startDate (uint256)

+

1620414600

_endDate (uint256)

+

1620415800

_limitWhale (uint256)

+

0

Write

View your transaction

2. antiWhaleActivated

True *bool*

7. limitWhale

0 *uint256*

6.4.3 Check if Wallet A with 1000 MEPAD Token is within the AntiWhale timeframe not able to sell or transfer

Send 100 MEPAD Token before AntiWhale timeframe started

Tx: <https://ropsten.etherscan.io/tx/0x070c7149f33ff41bc65b722e9fd26450660e31fefb90948a99f8f2f191680307>

Status: Success

Send 100 MEPAD Token within the AntiWhale timeframe

Tx: <https://ropsten.etherscan.io/tx/0x65247db995034d5825309d008462377f6ff2dca9309120ef856dedc93d1e83ed>

Status: Failed

? Status:

✖ Fail with error 'Error: No time for whales!'

? Block:

10192499

2 Block Confirmations

? Timestamp:

🕒 36 secs ago (May-07-2021 07:12:15 PM +UTC)

? From:

0xe1d80e7833f8b81d792922aecc0467ccfc3715bc 

? To:

Contract 0xbf7d39426f84ee43c5d62bacee1868b60b599213  

⌵ Warning! Error encountered during contract execution [Reverted] 😞

⌵ ⚠ ERC-20 Token Transfer Error (Unable to locate corresponding Transfer Event Logs), Check with Sender. 

6. isWhale

amount (uint256)

100


Query

⌵ bool

[isWhale(uint256) method Response]

➤ bool : true

6.5 Overall smart contract security needs to be checked

Status: tested and verified 



7. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The final debriefs took place on the May 07, 2021. The overall code quality of the project is good, not overloaded with unnecessary functions, these is greatly benefiting the security of the contract and also correctly implemented widely used and reviewed contracts from OpenZeppelin.

The main goal of the audit was to verify the claims regarding the security of the smart contract and the claims inside the scope of work. During the audit, no critical issues were found after the manual and automated security testing. The only concern within the contract was the AntiWhale function and good addressed (6.4) within our audit.

8. Deployed Smart Contract

VERIFIED

Smart Contract is deployed here:

<https://bscscan.com/address/0x9d70a3ee3079a6fa2bb16591414678b7ad91f0b5#code>

