



# CONTRACT WOLF

**Blockchain Security - Smart Contract Audits**

## Security Assessment

March 21, 2022





# Disclaimer

**ContractWolf.io** audits and reports should not be considered as a form of project's "advertisement" and does not cover any interaction and assessment from "project's contract" to "external contracts" such as Pancakeswap or similar.

**ContractWolf** does not provide any warranty on its released reports.

**ContractWolf** should not be used as a decision to invest into an audited project and is not affiliated nor partners to its audited contract projects.

**ContractWolf** provides transparent report to all its "clients" and to its "clients participants" and will not claim any guarantee of bug-free code within it's **SMART CONTRACT**.

**ContractWolf** presence is to analyze, audit and assess the client's smart contract's code.

Each company or projects should be liable to its security flaws and functionalities.

# Network

Binance Smart Chain (BEP20)

# Website

<https://meigma.finance>

# Telegram

<https://t.me/meigma>

# Twitter

[https://twitter.com/meigma\\_finance](https://twitter.com/meigma_finance)

## Description

**Meigma** is an Auto Staking and Compounding protocol boasting the most explosive fixed APY of 401,333.11%. It rewards users with an EXPLOSIVE APY that turns a \$1000 investment into over \$4,014,331 in just a year. Rewards are auto compounded and delivered to your wallet every 15 minutes, making it a massive 96 times per day. All you have to do is HODL to achieve Freedom and explosive rebases. Earnings are calculated in a scenario where the protocol sustains the rebase reward for 365 days.

2.5% of all \$MEIGMA traded are burnt in an exciting feature name "The Volcano Crater". The more that is traded, the more gets put into the crater causing the volcano crater to grow in size, larger and larger through self-fulfilling Auto-Compounding, reducing the circulating supply and keeping the token stable.

## ContractWolf Engagement

21<sup>st</sup> of March 2022, **Meigma** engaged and agrees to audit their smart contract's code by ContractWolf. The goal of this engagement was to identify if there is a possibility of security flaws in the implementation of the contract or system.

**ContractWolf** will be focusing on contract issues and functionalities along with the projects claims from smart contract to their website, whitepaper and repository which has been provided by **Meigma**.

## Logo



## Contract link

<https://bscscan.com/address/0x5f27fb302453f1403b29ec986919045c3d404f93>

# Risk Level Classification

Risk Level represents the classification or the probability that a certain function or threat that can exploit vulnerability and have an impact within the system or contract.

Risk Level is computed based on CVSS Version 3.0

Level	Value	Vulnerability
Critical	9 - 10	An Exposure that can affect the contract functions in several events that can risk and disrupt the contract
High	7 - 8.9	An Exposure that can affect the outcome when using the contract that can serve as an opening in manipulating the contract in an unwanted manner
Medium	4 - 6.9	An opening that could affect the outcome in executing the contract in a specific situation
Low	0.1 - 3.9	An opening but doesn't have an impact on the functionality of the contract
Informational	0	An opening that consists of information's but will not risk or affect the contract

# Auditing Approach

Every line of code along with its functionalities will undergo manual review to check its security issues, quality, and contract scope of inheritance. The manual review will be done by our team that will document any issues that there were discovered.

## Methodology

The auditing process follows a routine series of steps:

### 1. Code review that includes the following:

- Review of the specifications, sources, and instructions provided to ContractWolf to make sure we understand the size, scope, and functionality of the smart contract.
- Manual review of code, our team will have a process of reading the code line-by-line with the intention of identifying potential vulnerabilities and security flaws.

### 2. Testing and automated analysis that includes:

- Testing the smart contract functions with common test cases and scenarios, to ensure that it returns the expected results.

3. Best practices review, the team will review the contract with the aim to improve efficiency, effectiveness, clarifications, maintainability, security, and control within the smart contract.

4. Recommendations to help the project take steps to secure the smart contract.



# Used Code from other Frameworks/Smart Contracts (Direct Imports)

## Imported Packages

- ERC20Detailed
- IERC20
- IPancakeSwapFactory
- IPancakeSwapPair
- IPancakeSwapRouter
- Meigma
- Ownable
- SafeMath
- SafeMathInt

# Description

Optimization enabled: No

Version: v0.7.4

Decimal: 5

Symbol: MEIGMA

## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	2	4	1

### Exposed Functions

Version	Public	Private
1.0	10	-

Version	External	Internal
1.0	84	24

### State Variables

Version	Total	Public
1.0	39	27

## Capabilities

Version	Solidity Versions Observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	v0.7.4		Yes	Yes	No

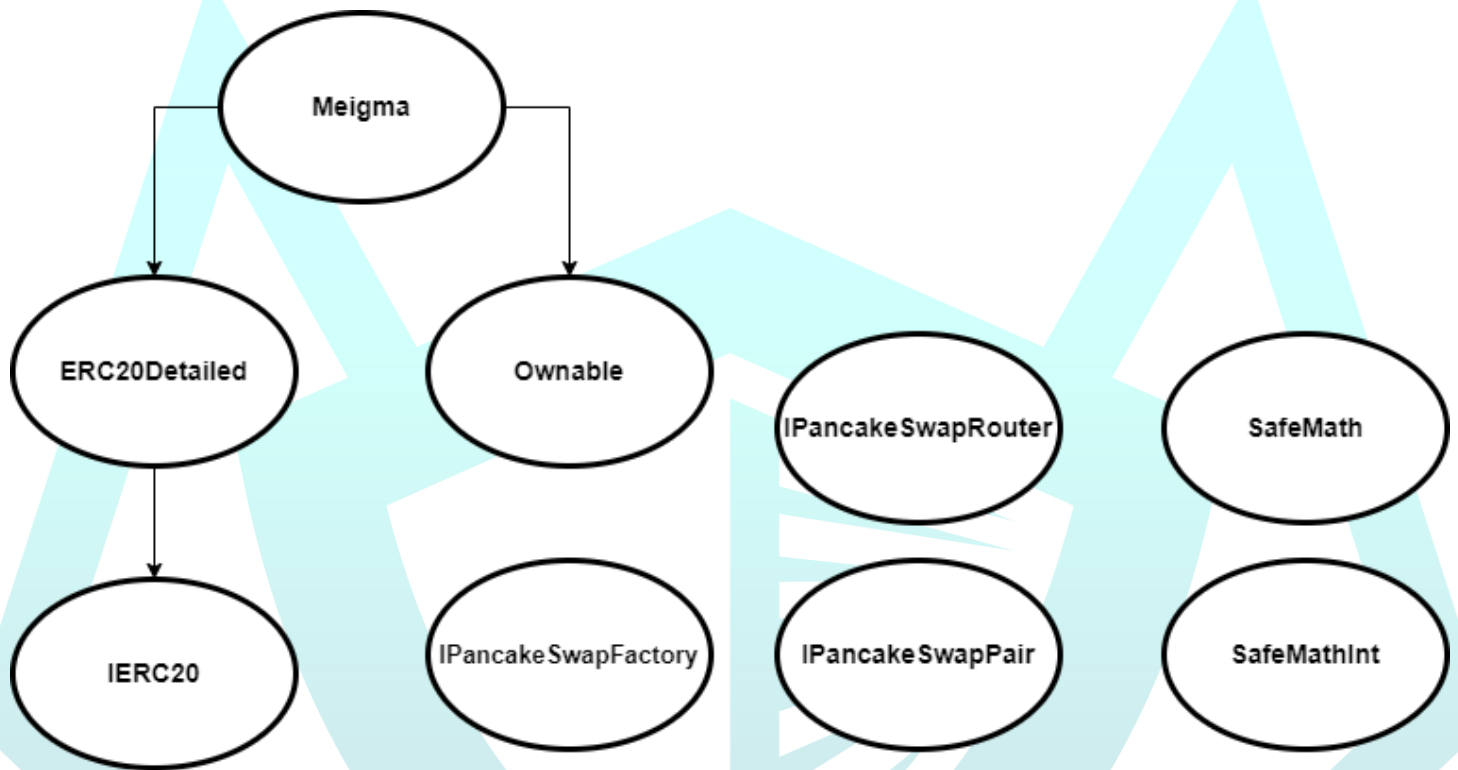


# Scope of Work

**Meigma's** team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract.



# Inheritance Graph



# Verify Claims

## Correct implementation of Token Standard

Tested	Verified
✓	✗

Function	Description	Exist	Tested	Verified
TotalSupply	Information about the total coin or token supply	✓	✓	✓
BalanceOf	Details on the account balance from a specified address	✓	✓	✓
Transfer	An action that transfers a specified amount of coin or token to a specified address	✓	✓	✓
TransferFrom	An action that transfers a specified amount of coin or token from a specified address	✓	✓	✓
Approve	Provides permission to withdraw specified number of coin or token from a specified address	✓	✓	✓

## Optional implementation

Function	Description	Exist	Tested	Verified
renounceOwnership	Owner renounce ownership for more trust	✓	✓	✓

## Deployer cannot mint after initial deployment

Statement	Exist	Tested	Verified	File
Deployer cannot mint	✓	✓	✓	Main

Max / Total supply: 325,000



### Deployer can block user

Statement	Exist	Tested	Verified
Deployer can block user	✓	✓	✓

### Deployer cannot burn

Statement	Exist	Tested	Verified
Deployer cannot burn	✓	✓	✓

### Deployer cannot pause contract

Statement	Exist	Tested	Verified
Deployer cannot pause	✓	✓	✓

# Overall Checkup (Smart Contract Security)

Tested	Verified
✓	✓

## Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	X
Unverified / Not checked	🚩
Not Available	—

# Write Functions of Contract

1. approve

2. decreaseAllowance

3. increaseAllowance

4. manualSync

5. renounceOwnership

6. setAutoAddLiquidity

7. setAutoRebase

8. setBotBlacklist

9. setFeeReceivers

10. setLP

11. setPairAddress

12. setWhitelist

13. transfer

14. transferFrom

15. transferOwnership

16. withdrawAllToTreasury



# SWC Attacks

ID	Title	Relationships	Status
<a href="#"><u>SWC-136</u></a>	Unencrypted Private Data On-Chain	<a href="#"><u>CWE-767: Access to Critical Private Variable via Public Method</u></a>	PASSED
<a href="#"><u>SWC-135</u></a>	Code With No Effects	<a href="#"><u>CWE-1164: Irrelevant Code</u></a>	PASSED
<a href="#"><u>SWC-134</u></a>	Message call with hardcoded gas amount	<a href="#"><u>CWE-655: Improper Initialization</u></a>	PASSED
<a href="#"><u>SWC-133</u></a>	Hash Collisions with Multiple Variable Length Arguments	<a href="#"><u>CWE-294: Authentication Bypass by Capture-replay</u></a>	PASSED
<a href="#"><u>SWC-132</u></a>	Unexpected Ether balance	<a href="#"><u>CWE-667: Improper Locking</u></a>	PASSED
<a href="#"><u>SWC-131</u></a>	Presence of unused variables	<a href="#"><u>CWE-1164: Irrelevant Code</u></a>	PASSED
<a href="#"><u>SWC-130</u></a>	Right-To Left Override control character (U+202E)	<a href="#"><u>CWE-451: User Interface (UI) Misrepresentation of Critical Information</u></a>	PASSED
<a href="#"><u>SWC-129</u></a>	Typographical Error	<a href="#"><u>CWE-480: Use of Incorrect Operator</u></a>	PASSED
<a href="#"><u>SWC-128</u></a>	DoS With Block Gas Limit	<a href="#"><u>CWE-400: Uncontrolled Resource Consumption</u></a>	PASSED

<a href="#"><u>SWC-127</u></a>	Arbitrary Jump with Function Type Variable	<a href="#"><u>CWE-695: Use of Low-Level Functionality</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-126</u></a>	Insufficient Gas Griefing	<a href="#"><u>CWE-691: Insufficient Control Flow Management</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-125</u></a>	Incorrect Inheritance Order	<a href="#"><u>CWE-696: Incorrect Behavior Order</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-124</u></a>	Write to Arbitrary Storage Location	<a href="#"><u>CWE-123: Write-what-where Condition</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-123</u></a>	Requirement Violation	<a href="#"><u>CWE-573: Improper Following of Specification by Caller</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-122</u></a>	Lack of Proper Signature Verification	<a href="#"><u>CWE-345: Insufficient Verification of Data Authenticity</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-121</u></a>	Missing Protection against Signature Replay Attacks	<a href="#"><u>CWE-347: Improper Verification of Cryptographic Signature</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-120</u></a>	Weak Sources of Randomness from Chain Attributes	<a href="#"><u>CWE-330: Use of Insufficiently Random Values</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-119</u></a>	Shadowing State Variables	<a href="#"><u>CWE-710: Improper Adherence to Coding Standards</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-118</u></a>	Incorrect Constructor Name	<a href="#"><u>CWE-665: Improper Initialization</u></a>	<b>PASSED</b>

<a href="#"><u>SWC-117</u></a>	Signature Malleability	<a href="#"><u>CWE-347: Improper Verification of Cryptographic Signature</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-116</u></a>	Timestamp Dependence	<a href="#"><u>CWE-829: Inclusion of Functionality from Untrusted Control Sphere</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-115</u></a>	Authorization through tx.origin	<a href="#"><u>CWE-477: Use of Obsolete Function</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-114</u></a>	Transaction Order Dependence	<a href="#"><u>CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-113</u></a>	DoS with Failed Call	<a href="#"><u>CWE-703: Improper Check or Handling of Exceptional Conditions</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-112</u></a>	Delegate call to Untrusted Callee	<a href="#"><u>CWE-829: Inclusion of Functionality from Untrusted Control Sphere</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-111</u></a>	Use of Deprecated Solidity Functions	<a href="#"><u>CWE-477: Use of Obsolete Function</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-110</u></a>	Assert Violation	<a href="#"><u>CWE-670: Always-Incorrect Control Flow Implementation</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-109</u></a>	Uninitialized Storage Pointer	<a href="#"><u>CWE-824: Access of Uninitialized Pointer</u></a>	<b>PASSED</b>

<a href="#"><u>SWC-108</u></a>	State Variable Default Visibility	<a href="#"><u>CWE-710: Improper Adherence to Coding Standards</u></a>	<b>NOT PASSED</b>
<a href="#"><u>SWC-107</u></a>	Reentrancy	<a href="#"><u>CWE-841: Improper Enforcement of Behavioral Workflow</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-106</u></a>	Unprotected SELFDESTRUCT Instruction	<a href="#"><u>CWE-284: Improper Access Control</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-105</u></a>	Unprotected Ether Withdrawal	<a href="#"><u>CWE-284: Improper Access Control</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-104</u></a>	Unchecked Call Return Value	<a href="#"><u>CWE-252: Unchecked Return Value</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-103</u></a>	Floating Pragma	<a href="#"><u>CWE-664: Improper Control of a Resource Through its Lifetime</u></a>	<b>NOT PASSED</b>
<a href="#"><u>SWC-102</u></a>	Outdated Compiler Version	<a href="#"><u>CWE-937: Using Components with Known Vulnerabilities</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-101</u></a>	Integer Overflow and Underflow	<a href="#"><u>CWE-682: Incorrect Calculation</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-100</u></a>	Function Default Visibility	<a href="#"><u>CWE-710: Improper Adherence to Coding Standards</u></a>	<b>PASSED</b>

# AUDIT PASSED

## Low Issues

A floating pragma is set (SWC – 103)	L: 58 C: 5
State variable visibility is not set (SWC – 108)	L: 447 C: 29, L: 472 C: 12, L: 473 C: 12, L: 483 C: 9



# Audit Comments

- Deployer cannot mint after initial deployment
- Deployer can block bot address
- Contract has a rebase variable
- Deployer cannot burn
- Deployer cannot set fees
- Deployer cannot pause contract





# CONTRACTWOLF

Blockchain Security - Smart Contract Audits