



CONTRACT WOLF

Blockchain Security - Smart Contract Audits

Security Assessment

January 14, 2022



REBORN

MetaSafeMoon

Disclaimer	3
Description	5
Engagement	5
Project Engagement	5
Logo	6
Contract Link	6
Risk Level Classification	7
Methodology	8
Used Code from other Frameworks / Smart Contracts (Imports)	9
Description	10
Scope of Work	12
Inheritance Graph	13
Verify Claim	14
Overall Checkup	20
Write Functions of Contract	21
SWC Attack	22
Audit Result	26
Audit Comments	27

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

BSC / Binance Smart Chain (BEP20 protocol)

Website

<https://metasafemoon.net/>

Twitter

<https://twitter.com/MetaSafeMoon>

Telegram

<https://t.me/MetaSafeMoonOfficial>

Other channels

Reddit: <https://www.reddit.com/r/MetaSafeMoon/>

Description

MetaSafeMoon is a BEP20 token designed to fulfill one of the first desires of the modern mankind, life outside earth.

MetaSafeMoon operates in an ecosystem that has carefully combined Metaverse and NFTs for a single purpose, offering a real utility that brings users from all blockchains in the same direction

ContractWolf Engagement

14th of January 2022, **MetaSafeMoon** 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 **MetaSafeMoon**.

Logo



Contract Link

<https://bscscan.com/address/0xf7a086bff67ded4aa785e8a0a81d4345d9bb4740>

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

- IERC20
- SafeMath
- Ownable
- InterfaceLP
- ERC20Detailed
- SafeMathInt
- IDEXFactory
- IDEXRouter
- METASFM

Description

Optimization enabled: Yes

Version: v0.7.4

Decimals: 9

Symbol: \$METASFM

Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	2	4	1

Exposed Functions

Version	Public	Private
1.0	11	1

Version	External	Internal
1.0	44	19

State Variables

Version	Total	Public
1.0	28	15

Capabilities

Version	Solidity Versions Observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	^0.7.4		Yes		No

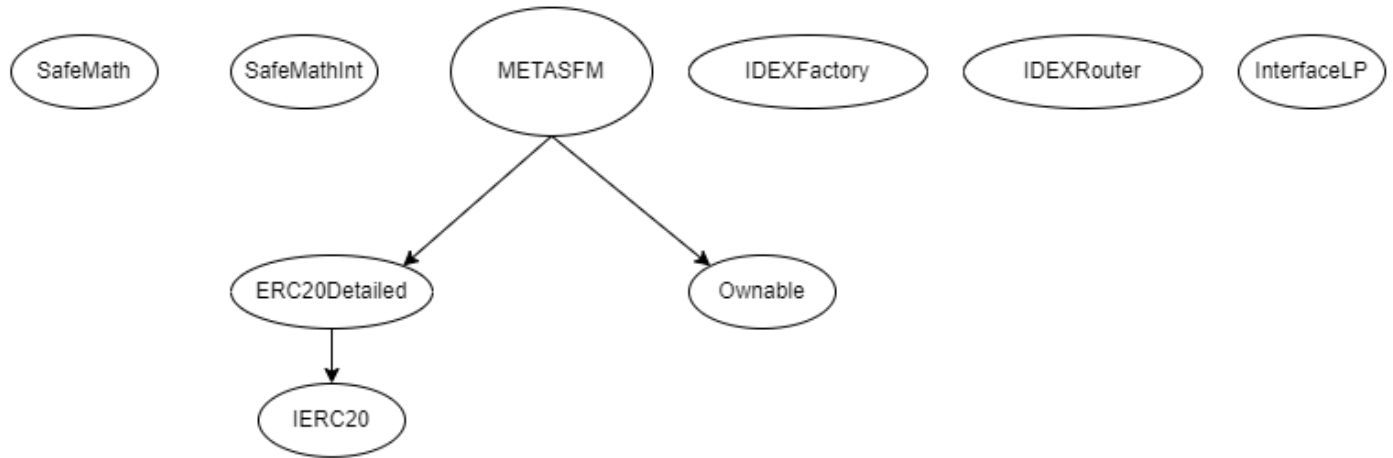


Scope of Work

MetaSafeMoon'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	✓	✓	✓
Allowance	Sets a specific number of coin or token that allows a specified address to utilize	✓	✓	✓

Optional implementation

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

Deployer cannot mint any new tokens

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

Max / Total supply: 1,000,000,000,000,000

Deployer cannot pause user funds

Statement	Exist	Tested	Verified
Deployer cannot pause	✓	✓	✓

Deployer cannot burn user funds

Statement	Exist	Tested	Verified
Deployer cannot burn	✓	✓	✓

Deployer cannot pause the 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. clearStuckBalance

3. decreaseAllowance

4. enableTransfer

5. increaseAllowance

6. manualSync

7. rebase

8. renounceOwnership

9. rescueToken

10. sendPresale

11. setFeeExempt

12. setFeeReceivers

13. setFees

14. setInitialDistributionFinished

15. setLP

16. setMaster

17. setMaxWalletExempt

18. setMaxWalletToken

19. setSwapBackSettings

20. setTargetLiquidity

21. transfer

22. transferFrom

23. transferOwnership

SWC Attacks

ID	Title	Relationships	Status
<u>SWC-136</u>	Unencrypted Private Data On-Chain	<u>CWE-767: Access to Critical Private Variable via Public Method</u>	PASSED
<u>SWC-135</u>	Code With No Effects	<u>CWE-1164: Irrelevant Code</u>	NOT PASSED
<u>SWC-134</u>	Message call with hardcoded gas amount	<u>CWE-655: Improper Initialization</u>	PASSED
<u>SWC-133</u>	Hash Collisions with Multiple Variable Length Arguments	<u>CWE-294: Authentication Bypass by Capture-replay</u>	PASSED
<u>SWC-132</u>	Unexpected Ether balance	<u>CWE-667: Improper Locking</u>	PASSED
<u>SWC-131</u>	Presence of unused variables	<u>CWE-1164: Irrelevant Code</u>	PASSED
<u>SWC-130</u>	Right-To Left Override control character (U+202E)	<u>CWE-451: User Interface (UI) Misrepresentation of Critical Information</u>	PASSED
<u>SWC-129</u>	Typographical Error	<u>CWE-480: Use of Incorrect Operator</u>	PASSED
<u>SWC-128</u>	DoS With Block Gas Limit	<u>CWE-400: Uncontrolled Resource Consumption</u>	PASSED
<u>SWC-127</u>	Arbitrary Jump with Function Type Variable	<u>CWE-695: Use of Low-Level Functionality</u>	PASSED

<u>SWC-125</u>	Incorrect Inheritance Order	<u>CWE-696: Incorrect Behavior Order</u>	PASSED
<u>SWC-124</u>	Write to Arbitrary Storage Location	<u>CWE-123: Write-what-where Condition</u>	PASSED
<u>SWC-123</u>	Requirement Violation	<u>CWE-573: Improper Following of Specification by Caller</u>	PASSED
<u>SWC-122</u>	Lack of Proper Signature Verification	<u>CWE-345: Insufficient Verification of Data Authenticity</u>	PASSED
<u>SWC-121</u>	Missing Protection against Signature Replay Attacks	<u>CWE-347: Improper Verification of Cryptographic Signature</u>	PASSED
<u>SWC-120</u>	Weak Sources of Randomness from Chain Attributes	<u>CWE-330: Use of Insufficiently Random Values</u>	PASSED
<u>SWC-119</u>	Shadowing State Variables	<u>CWE-710: Improper Adherence to Coding Standards</u>	PASSED
<u>SWC-118</u>	Incorrect Constructor Name	<u>CWE-665: Improper Initialization</u>	PASSED
<u>SWC-117</u>	Signature Malleability	<u>CWE-347: Improper Verification of Cryptographic Signature</u>	PASSED
<u>SWC-116</u>	Timestamp Dependence	<u>CWE-829: Inclusion of</u>	PASSED

		<u>Functionality from Untrusted Control Sphere</u>	
<u>SWC-115</u>	Authorization through tx.origin	<u>CWE-477: Use of Obsolete Function</u>	PASSED
<u>SWC-114</u>	Transaction Order Dependence	<u>CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</u>	PASSED
<u>SWC-113</u>	DoS with Failed Call	<u>CWE-703: Improper Check or Handling of Exceptional Conditions</u>	PASSED
<u>SWC-112</u>	Delegate call to Untrusted Callee	<u>CWE-829: Inclusion of Functionality from Untrusted Control Sphere</u>	PASSED
<u>SWC-111</u>	Use of Deprecated Solidity Functions	<u>CWE-477: Use of Obsolete Function</u>	PASSED
<u>SWC-110</u>	Assert Violation	<u>CWE-670: Always-Incorrect Control Flow Implementation</u>	PASSED
<u>SWC-109</u>	Uninitialized Storage Pointer	<u>CWE-824: Access of Uninitialized Pointer</u>	PASSED
<u>SWC-108</u>	State Variable Default Visibility	<u>CWE-710: Improper Adherence to Coding Standards</u>	PASSED

<u>SWC-107</u>	Reentrancy	<u>CWE-841: Improper Enforcement of Behavioral Workflow</u>	PASSED
<u>SWC-106</u>	Unprotected SELFDESTRUCT Instruction	<u>CWE-284: Improper Access Control</u>	PASSED
<u>SWC-105</u>	Unprotected Ether Withdrawal	<u>CWE-284: Improper Access Control</u>	PASSED
<u>SWC-104</u>	Unchecked Call Return Value	<u>CWE-252: Unchecked Return Value</u>	PASSED
<u>SWC-103</u>	Floating Pragma	<u>CWE-664: Improper Control of a Resource Through its Lifetime</u>	PASSED
<u>SWC-102</u>	Outdated Compiler Version	<u>CWE-937: Using Components with Known Vulnerabilities</u>	PASSED
<u>SWC-101</u>	Integer Overflow and Underflow	<u>CWE-682: Incorrect Calculation</u>	PASSED
<u>SWC-100</u>	Function Default Visibility	<u>CWE-710: Improper Adherence to Coding Standards</u>	PASSED

AUDIT PASSED

Critical Issues

No critical issues found

High Issues

No high issues found

Medium Issues

No medium issues found

Low Issues

No low issues found

Informational Issues

No informational issues found

Function Issues

No informational issues found

Audit Comments

January 17, 2022

- Too many external addresses declared
- Irrelevant line of code 367

