Mashida

Smart Contract Audit Report

SYSFIXED

15 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Mashida	MSHD	BSC

Addresses

Contract address	0x06CE168FF4Ca760768f42C440d4266BA705E2F21
Contract deployer address	0x5ab868961f4F18C553BdA8CE61cB338E5E1bB300

Project Website

https://mashida.io/

Codebase

https://bscscan.com/address/0x06CE168FF4Ca760768f42C440d4266BA705E2F21#contracts

SUMMARY

MASHIDA is a BEP-20 token built on the BNB blockchain , it is a Crypto Token and a Web3 Platform that contains a Virtual world, Social and Game application features that are interconnected, here people can interact virtually, work, play, and meet based on activity and interaction, they can transact peer to peer. Application owners and users are referred to as the Mashida Army, with NFT as profile identities and assets on the platform, $MSHD will be required for buying and selling non-fungible tokens (NFTs).

Contract Summary

Documentation Quality

Mashida provides a very good documentation with standard of solidity base code.

- The technical description is provided clearly and structured and also dont have any high risk issue.

Code Quality

The Overall quality of the basecode is standard.

- Standart solidity basecode and rules are already followed with Mashida with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-100 SWC-108 | Explicitly define visibility for all state variables on lines 44.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 114, 115, 116, 117, 118, 119, 120, 155, 163, 170, 189, 191, 200 and 201.

CONCLUSION

We have audited the Mashida project which has released on January 2023 to discover issues and identify potential security vulnerabilities in Mashida Project. This process is used to find technical issues and security loopholes that find some common issues in the code.

The security audit report produced satisfactory results with low-risk issues.

The most common issue found in writing code on contracts that do not pose a big risk, writing on contracts is close to the standard of writing contracts in general. The low-level issues found are some arithmetic operation issues and a state variable visibility is not set. It is best practice to set the visibility of state variables explicitly.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	PASS
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	PASS
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Caller	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS

Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS

SMART CONTRACT ANALYSIS

Started	Sat Jan 14 2023 21:27:40 GMT+0000 (Coordinated Universal Time)
Finished	Sun Jan 15 2023 00:21:20 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Mashida.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged

SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 114

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Mashida.sol

Locations

```
113   //split TGE
114   _mint(PRESALE, 100000000 * 10 ** uint256(_decimals)); //1%
115   _mint(LIQUIDITY_PROVISION, 3200000000 * 10 ** uint256(_decimals)); //32%
116   _mint(ECOSYSTEM, 2000000000 * 10 ** uint256(_decimals)); //20%
117   _mint(TEAM, 1000000000 * 10 ** uint256(_decimals)); //10%
118
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 115

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Mashida.sol

Locations

```
114   _mint(PRESALE, 100000000 * 10 ** uint256(_decimals)); //1%
115   _mint(LIQUIDITY_PROVISION, 3200000000 * 10 ** uint256(_decimals)); //32%
116   _mint(ECOSYSTEM, 2000000000 * 10 ** uint256(_decimals)); //20%
117   _mint(TEAM, 1000000000 * 10 ** uint256(_decimals)); //10%
118   _mint(MARKETING, 1500000000 * 10 ** uint256(_decimals)); //15%
119
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 116

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Mashida.sol

Locations

```
115   _mint(LIQUIDITY_PROVISION, 3200000000 * 10 ** uint256(_decimals)); //32%
116   _mint(ECOSYSTEM, 2000000000 * 10 ** uint256(_decimals)); //20%
117   _mint(TEAM, 1000000000 * 10 ** uint256(_decimals)); //10%
118   _mint(MARKETING, 1500000000 * 10 ** uint256(_decimals)); //15%
119   _mint(PRODUCT_DEVELOPMENT, 1500000000 * 10 ** uint256(_decimals)); //15%
120
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 117

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Mashida.sol

Locations

```
116   _mint(ECOSYSTEM, 2000000000 * 10 ** uint256(_decimals)); //20%
117   _mint(TEAM, 1000000000 * 10 ** uint256(_decimals)); //10%
118   _mint(MARKETING, 1500000000 * 10 ** uint256(_decimals)); //15%
119   _mint(PRODUCT_DEVELOPMENT, 1500000000 * 10 ** uint256(_decimals)); //15%
120   _mint(TREASURY, 700000000 * 10 ** uint256(_decimals)); //7%
121
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 118

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Mashida.sol

Locations

```
117   _mint(TEAM, 1000000000 * 10 ** uint256(_decimals)); //10%
118   _mint(MARKETING, 1500000000 * 10 ** uint256(_decimals)); //15%
119   _mint(PRODUCT_DEVELOPMENT, 1500000000 * 10 ** uint256(_decimals)); //15%
120   _mint(TREASURY, 700000000 * 10 ** uint256(_decimals)); //7%
121   pinkAntiBot = IPinkAntiBot(pinkAntiBot_);
122
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 119

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Mashida.sol

Locations

```
118   _mint(MARKETING, 1500000000 * 10 ** uint256(_decimals)); //15%
119   _mint(PRODUCT_DEVELOPMENT, 1500000000 * 10 ** uint256(_decimals)); //15%
120   _mint(TREASURY, 700000000 * 10 ** uint256(_decimals)); //7%
121   pinkAntiBot = IPinkAntiBot(pinkAntiBot_);
122   pinkAntiBot.setTokenOwner(msg.sender);
123
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 120

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Mashida.sol

Locations

```
119   _mint(PRODUCT_DEVELOPMENT, 1500000000 * 10 ** uint256(_decimals)); //15%
120   _mint(TREASURY, 700000000 * 10 ** uint256(_decimals)); //7%
121   pinkAntiBot = IPinkAntiBot(pinkAntiBot_);
122   pinkAntiBot.setTokenOwner(msg.sender);
123   antiBotEnabled = true;
124
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 155

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Mashida.sol

Locations

```
154   if(_allowances[sender][_msgSender()] != ~uint(0)){
155   _allowances[sender][_msgSender()] = _allowances[sender][_msgSender()]-(amount);
156   }
157
158   _transfer(sender, recipient, amount);
159
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 163

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Mashida.sol

Locations

```
162   function increaseAllowance(address spender, uint256 addedValue) public virtual
returns (bool) {
163   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
164   return true;
165   }
166   function decreaseAllowance(address spender, uint256 subtractedValue) public virtual
returns (bool) {
167
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 170

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Mashida.sol

Locations

```
169   unchecked {
170   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
171   }
172   return true;
173   }
174
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 189

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Mashida.sol

Locations

```
188   }
189   _balances[sender] = senderBalance - amount;
190
191   _balances[recipient] += amount;
192   emit Transfer(sender, recipient, amount);
193
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 191

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Mashida.sol

Locations

```
190
191   _balances[recipient] += amount;
192   emit Transfer(sender, recipient, amount);
193   return true;
194   }
195
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 200

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Mashida.sol

Locations

```
199
200   _totalSupply += amount;
201   _balances[account] += amount;
202   emit Transfer(address(0), account, amount);
203   }
204
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 201

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- Mashida.sol

Locations

```
200   _totalSupply += amount;
201   _balances[account] += amount;
202   emit Transfer(address(0), account, amount);
203   }
204
205
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 44

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_creator" is internal. Other possible visibility settings are public and private.

Source File

- Mashida.sol

Locations

```
43   address public _owner;
44   address immutable _creator;
45
46   event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
47   constructor() {
48
```

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to, or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Sysfixed's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Sysfixed to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model, or legal compliance.

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.