JOJO

Smart Contract Audit Report

SYSFIXED

27 Jul 2021

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
JOJO	JOJO	Binance Smart Chain

Addresses

Contract address	0x78a499a998bdd5a84cf8b5abe49100d82de12f1c
Contract deployer address	0x5A8097188219D015412EA2cBdd7662CCb29aE5BC

Project Website

https://jojo.fun/home

Codebase

https://bscscan.com/address/0x78a499a998bdd5a84cf8b5abe49100d82de12f1c#code

SUMMARY

JOJO is a NFT Metaverse project that is about to run on BSC. JOJO combines MEME, NFT, Metaverse and SmartTOY to build a world-class pan-entertainment platform.

Contract Summary

Documentation Quality

JOJO 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.

- Standard solidity basecode and rules are already followed by JOJO 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 806.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 111, 123, 136, 137, 148, 158, 172, 189, 204, 205, 223, 240, 258, 278, 298, 781, 781, 781, 781, 782, 782, 810, 810, 810, 810, 811, 811, 811, 811, 850, 850, 946, 948, 1123, 1129, 1135, 1170, 1277 and 948.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 9, 89, 306, 333, 403, 593, 692, 738 and 758.
- SWC-110 SWC-123 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 947, 948, 948, 1091, 1092, 1171, 1171, 1172 and 1173.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 850 and 1022.

CONCLUSION

We have audited the JOJO project released on July 2021 to discover issues and identify potential security vulnerabilities in JOJO Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

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

The issues found in the JOJO smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, a state variable visibility is not set, the potential use of "block.number" as a source of randomness, and out-of-bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value. The current pragma Solidity directive is "">=0.6.00.8.0"". Specifying a fixed compiler version is recommended to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code. The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number, and timestamp are predictable and can be manipulated by a malicious miner. Also, keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness, and be aware that using these variables introduces a certain level of trust in miners.

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.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	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
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	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
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	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.	ISSUE FOUND
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	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
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

Typographical Error	SWC-129	A typographical error can occur for example when the intent of a defined operation is to sum a number to a variable.	PASS
Override control character	SWC-130	Malicious actors can use the Right-To-Left-Override unicode character to force RTL text rendering and confuse users as to the real intent of a contract.	PASS
Unused variables	SWC-131 SWC-135	Unused variables are allowed in Solidity and they do not pose a direct security issue.	PASS
Unexpected Ether balance	SWC-132	Contracts can behave erroneously when they strictly assume a specific Ether balance.	PASS
Hash Collisions Variable	SWC-133	Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The transfer() and send() functions forward a fixed amount of 2300 gas.	PASS
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	PASS

SMART CONTRACT ANALYSIS

Started	Monday Jul 26 2021 05:16:06 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Jul 27 2021 08:18:51 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	JOJO.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-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-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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged

SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 111

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
110   function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint256) {
111   uint256 c = a + b;
112   if (c < a) return (false, 0);
113   return (true, c);
114   }
115
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 123

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
122   if (b > a) return (false, 0);
123   return (true, a - b);
124   }
125
126   /**
127
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 136

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
135   if (a == 0) return (true, 0);
136   uint256 c = a * b;
137   if (c / a != b) return (false, 0);
138   return (true, c);
139   }
140
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 137

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
136   uint256 c = a * b;
137   if (c / a != b) return (false, 0);
138   return (true, c);
139   }
140
141
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 148

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
147   if (b == 0) return (false, 0);
148   return (true, a / b);
149   }
150
151   /**
152
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 158

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
157   if (b == 0) return (false, 0);
158   return (true, a % b);
159   }
160
161   /**
162
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 172

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
171   function add(uint256 a, uint256 b) internal pure returns (uint256) {
172   uint256 c = a + b;
173   require(c >= a, "SafeMath: addition overflow");
174   return c;
175   }
176
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 189

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
188   require(b <= a, "SafeMath: subtraction overflow");
189   return a - b;
190   }
191
192   /**
193
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 204

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
203   if (a == 0) return 0;
204   uint256 c = a * b;
205   require(c / a == b, "SafeMath: multiplication overflow");
206   return c;
207   }
208
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 205

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
204   uint256 c = a * b;
205   require(c / a == b, "SafeMath: multiplication overflow");
206   return c;
207   }
208
209
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 223

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
222   require(b > 0, "SafeMath: division by zero");
223   return a / b;
224   }
225
226   /**
227
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 240

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
239   require(b > 0, "SafeMath: modulo by zero");
240   return a % b;
241   }
242
243   /**
244
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 258

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
257   require(b <= a, errorMessage);
258   return a - b;
259   }
260
261   /**
262
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 278

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
277   require(b > 0, errorMessage);
278   return a / b;
279   }
280
281   /**
282
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 298

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
297   require(b > 0, errorMessage);
298   return a % b;
299   }
300   }
301
302
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 781

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
780   uint256 private constant MAX = ~uint256(0);
781   uint256 private _tTotal = 1000000 * 10**6 * 10**9;
782   uint256 private _rTotal = (MAX - (MAX % _tTotal));
783   uint256 private _tFeeTotal;
784   uint256 private _tPoolTotal;
785
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 781

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
780   uint256 private constant MAX = ~uint256(0);
781   uint256 private _tTotal = 1000000 * 10**6 * 10**9;
782   uint256 private _rTotal = (MAX - (MAX % _tTotal));
783   uint256 private _tFeeTotal;
784   uint256 private _tPoolTotal;
785
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 781

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
780   uint256 private constant MAX = ~uint256(0);
781   uint256 private _tTotal = 1000000 * 10**6 * 10**9;
782   uint256 private _rTotal = (MAX - (MAX % _tTotal));
783   uint256 private _tFeeTotal;
784   uint256 private _tPoolTotal;
785
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 781

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
780   uint256 private constant MAX = ~uint256(0);
781   uint256 private _tTotal = 1000000 * 10**6 * 10**9;
782   uint256 private _rTotal = (MAX - (MAX % _tTotal));
783   uint256 private _tFeeTotal;
784   uint256 private _tPoolTotal;
785
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 782

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
781   uint256 private _tTotal = 1000000 * 10**6 * 10**9;
782   uint256 private _rTotal = (MAX - (MAX % _tTotal));
783   uint256 private _tFeeTotal;
784   uint256 private _tPoolTotal;
785
786
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 782

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
781   uint256 private _tTotal = 1000000 * 10**6 * 10**9;
782   uint256 private _rTotal = (MAX - (MAX % _tTotal));
783   uint256 private _tFeeTotal;
784   uint256 private _tPoolTotal;
785
786
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 810

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
809   // 100e
810   uint256 public _maxTxAmount = 10000 * 10**6 * 10**9;
811   uint256 public numTokensSellToAddToLiquidity = 500 * 10**6 * 10**9;
812
813   // Prevent front run buy
814
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 810

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
809   // 100e
810   uint256 public _maxTxAmount = 10000 * 10**6 * 10**9;
811   uint256 public numTokensSellToAddToLiquidity = 500 * 10**6 * 10**9;
812
813   // Prevent front run buy
814
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 810

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
809   // 100e
810   uint256 public _maxTxAmount = 10000 * 10**6 * 10**9;
811   uint256 public numTokensSellToAddToLiquidity = 500 * 10**6 * 10**9;
812
813   // Prevent front run buy
814
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 810

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
809   // 100e
810   uint256 public _maxTxAmount = 10000 * 10**6 * 10**9;
811   uint256 public numTokensSellToAddToLiquidity = 500 * 10**6 * 10**9;
812
813   // Prevent front run buy
814
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 811

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
810   uint256 public _maxTxAmount = 10000 * 10**6 * 10**9;
811   uint256 public numTokensSellToAddToLiquidity = 500 * 10**6 * 10**9;
812
813   // Prevent front run buy
814   uint256 public startBuyBlock;
815
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 811

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
810   uint256 public _maxTxAmount = 10000 * 10**6 * 10**9;
811   uint256 public numTokensSellToAddToLiquidity = 500 * 10**6 * 10**9;
812
813   // Prevent front run buy
814   uint256 public startBuyBlock;
815
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 811

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
810   uint256 public _maxTxAmount = 10000 * 10**6 * 10**9;
811   uint256 public numTokensSellToAddToLiquidity = 500 * 10**6 * 10**9;
812
813   // Prevent front run buy
814   uint256 public startBuyBlock;
815
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 811

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
810   uint256 public _maxTxAmount = 10000 * 10**6 * 10**9;
811   uint256 public numTokensSellToAddToLiquidity = 500 * 10**6 * 10**9;
812
813   // Prevent front run buy
814   uint256 public startBuyBlock;
815
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 850

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
849
850   startBuyBlock = block.number + 20 * 15;
851   emit Transfer(address(0), _msgSender(), _tTotal);
852   }
853
854
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 850

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
849
850   startBuyBlock = block.number + 20 * 15;
851   emit Transfer(address(0), _msgSender(), _tTotal);
852   }
853
854
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 946

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
945   require(_isExcluded[account], "Account is not excluded");
946   for (uint256 i = 0; i < _excluded.length; i++) {
947   if (_excluded[i] == account) {
948   _excluded[i] = _excluded[_excluded.length - 1];
949   _tOwned[account] = 0;
950
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 948

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
947   if (_excluded[i] == account) {
948   _excluded[i] = _excluded[_excluded.length - 1];
949   _tOwned[account] = 0;
950   _isExcluded[account] = false;
951   _excluded.pop();
952
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1123

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
1122   return _amount.mul(_taxFee).div(
1123   10**2
1124   );
1125   }
1126
1127
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1129

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
1128   return _amount.mul(_liquidityFee).div(
1129   10**2
1130   );
1131   }
1132
1133
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1135

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
1134   return _amount.mul(_poolFee).div(
1135   10**2
1136   );
1137   }
1138
1139
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1170

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
1169   uint256 tSupply = _tTotal;
1170   for (uint256 i = 0; i < _excluded.length; i++) {
1171   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
1172   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1173   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1174
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1277

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
1276   _maxTxAmount = _tTotal.mul(maxTxPercent).div(
1277   10**2
1278   );
1279   }
1280
1281
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 948

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
947   if (_excluded[i] == account) {
948   _excluded[i] = _excluded[_excluded.length - 1];
949   _tOwned[account] = 0;
950   _isExcluded[account] = false;
951   _excluded.pop();
952
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 9

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- JOJO.sol

Locations

```
8
9   pragma solidity >=0.6.0 <0.8.0;
10
11   /**
12   * @dev Interface of the ERC20 standard as defined in the EIP.
13
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 89

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- JOJO.sol

Locations

```
88
89   pragma solidity >=0.6.0 <0.8.0;
90
91   /**
92   * @dev Wrappers over Solidity's arithmetic operations with added overflow
93
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 306

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- JOJO.sol

Locations

```
305
306   pragma solidity >=0.6.0 <0.8.0;
307
308   /*
309   * @dev Provides information about the current execution context, including the
310
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 333

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- JOJO.sol

Locations

```
332
333   pragma solidity >=0.6.0 <0.8.0;
334
335   /**
336   * @dev Contract module which provides a basic access control mechanism, where
337
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 403

low SEVERITY

The current pragma Solidity directive is ""=0.6.2<0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- JOJO.sol

Locations

```
402
403   pragma solidity >=0.6.2 <0.8.0;
404
405   /**
406   * @dev Collection of functions related to the address type
407
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 593

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- JOJO.sol

Locations

```
592
593   pragma solidity ^0.6.2;
594
595
596   interface IUniswapV2Router01 {
597
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 692

low SEVERITY

The current pragma Solidity directive is ""=0.6.2"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- JOJO.sol

Locations

```
691
692   pragma solidity >=0.6.2;
693
694
695   interface IUniswapV2Router02 is IUniswapV2Router01 {
696
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 738

low SEVERITY

The current pragma Solidity directive is ""^0.5.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- JOJO.sol

Locations

```
737
738   pragma solidity ^0.5.0;
739
740   interface IUniswapV2Factory {
741
742
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 758

low SEVERITY

The current pragma Solidity directive is ""^0.6.12"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- JOJO.sol

Locations

```
757
758   pragma solidity ^0.6.12;
759
760
761
762
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 806

low SEVERITY

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

Source File

- JOJO.sol

Locations

```
805
806   bool inSwapAndLiquify;
807   bool public swapAndLiquifyEnabled = true;
808
809   // 100e
810
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 947

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- JOJO.sol

Locations

```
946   for (uint256 i = 0; i < _excluded.length; i++) {
947   if (_excluded[i] == account) {
948   _excluded[i] = _excluded[_excluded.length - 1];
949   _tOwned[account] = 0;
950   _isExcluded[account] = false;
951
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 948

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- JOJO.sol

Locations

```
947   if (_excluded[i] == account) {
948   _excluded[i] = _excluded[_excluded.length - 1];
949   _tOwned[account] = 0;
950   _isExcluded[account] = false;
951   _excluded.pop();
952
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 948

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- JOJO.sol

Locations

```
947   if (_excluded[i] == account) {
948   _excluded[i] = _excluded[_excluded.length - 1];
949   _tOwned[account] = 0;
950   _isExcluded[account] = false;
951   _excluded.pop();
952
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1091

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- JOJO.sol

Locations

```
1090   address[] memory path = new address[](2);
1091   path[0] = address(this);
1092   path[1] = uniswapV2Router.WETH();
1093
1094   _approve(address(this), address(uniswapV2Router), tokenAmount);
1095
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1092

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- JOJO.sol

Locations

```
1091   path[0] = address(this);
1092   path[1] = uniswapV2Router.WETH();
1093
1094   _approve(address(this), address(uniswapV2Router), tokenAmount);
1095
1096
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1171

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- JOJO.sol

Locations

```
1170   for (uint256 i = 0; i < _excluded.length; i++) {
1171   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
1172   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1173   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1174   }
1175
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1171

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- JOJO.sol

Locations

```
1170   for (uint256 i = 0; i < _excluded.length; i++) {
1171   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
1172   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1173   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1174   }
1175
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1172

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- JOJO.sol

Locations

```
1171   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
1172   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1173   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1174   }
1175   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1176
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1173

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- JOJO.sol

Locations

```
1172   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1173   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1174   }
1175   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1176   return (rSupply, tSupply);
1177
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 850

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- JOJO.sol

Locations

```
849
850   startBuyBlock = block.number + 20 * 15;
851   emit Transfer(address(0), _msgSender(), _tTotal);
852   }
853
854
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 1022

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- JOJO.sol

Locations

```
1021   // Prevent front run buy, uniswap, When the purchase has not started
1022   if(from == uniswapV2Pair && block.number < startBuyBlock){
1023   revert("the purchase has not started");
1024   }
1025   }
1026
```

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.