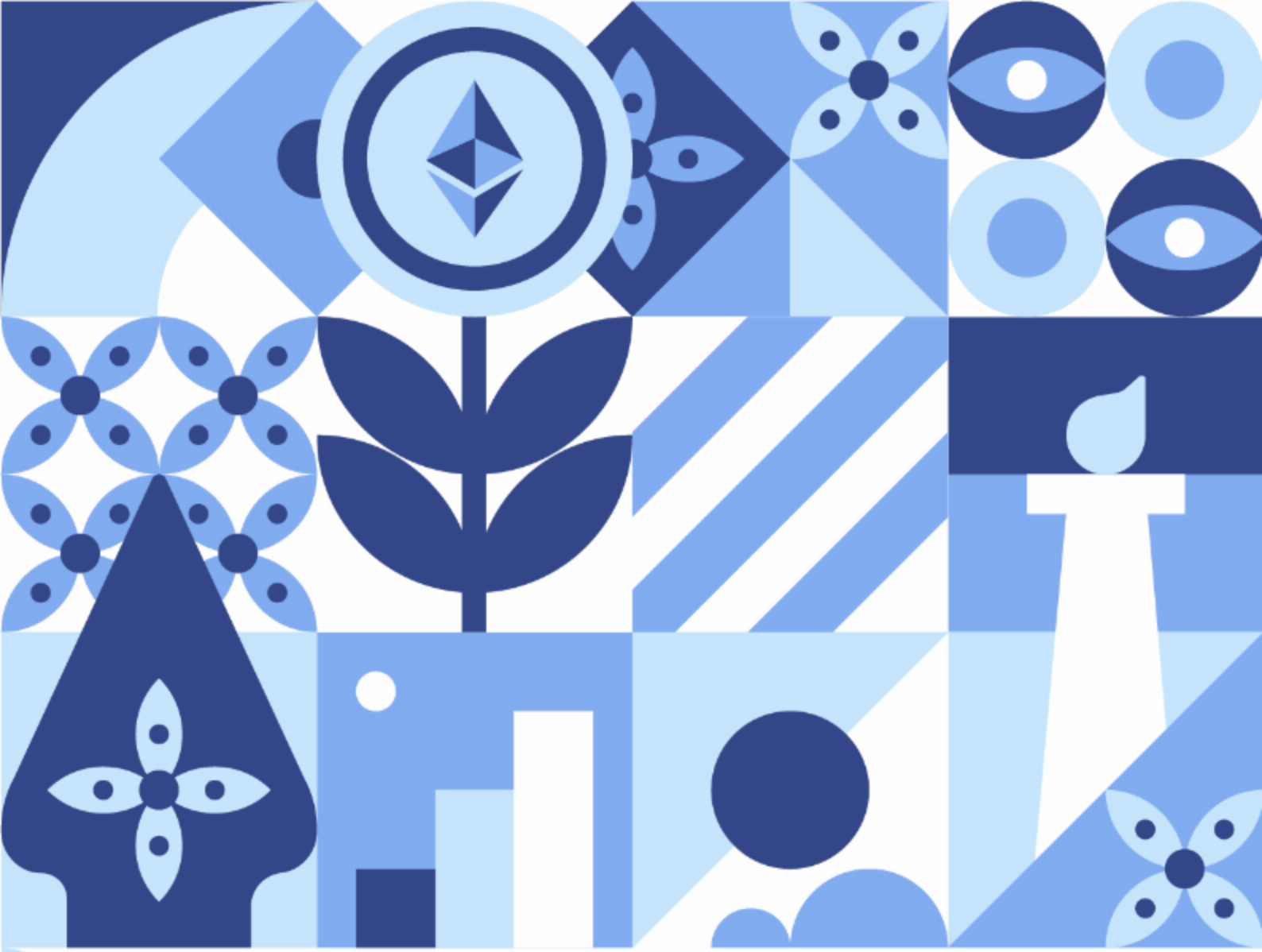

JigsawToken

Smart Contract Audit Report

SYSFIXED

15 Jul 2022

TABLE OF CONTENTS

Audited Details

- Audited Project
- Blockchain
- Addresses
- Project Website
- Codebase

Summary

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

Conclusion

Audit Results

Smart Contract Analysis

- Detected Vulnerabilities

Disclaimer

About Us

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
JigsawToken	JIGSAW	Ethereum

Addresses

Contract address	0xb0d47dD82fb8FACb1Bc4bA534a836B545aD97d2B
Contract deployer address	0xe117BCA4647B832Aa2fbE54031BA53C327b5bA45

Project Website

https://jigsawtoken.net/

Codebase

https://etherscan.io/address/0xb0d47dD82fb8FACb1Bc4bA534a836B545aD97d2B#code

SUMMARY

Jigsaw intends to provide an environment that is beneficial to the holder and to all communities. As you will see, our theme may skirt horror, thriller, and suspense, but our intent is on legitimacy, transparency, and profitability for both the individual and DeFi community.

Contract Summary

Documentation Quality

JigsawToken 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 JigsawToken with the discovery of several low issues.

Test Coverage

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

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 97, 109, 122, 123, 134, 146, 197, 383, 383, 386, 386, 389, 389, 502, 605, 630, 630, 631, 646, 665, 665, 717, 717, 717, 718, 722, 723, 723, 725, 725, 725, 726, 726, 727, 727 and 197.
- 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 170, 198, 203, 579, 741 and 742.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 475 and 646.

CONCLUSION

We have audited the JigsawToken project released on July 2022 to discover issues and identify potential security vulnerabilities in JigsawToken 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 a satisfactory result with some low-risk issues.

The issues found in the JigsawToken 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, weak sources 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. We recommend to don't using any of those environment variables as sources of randomness and being aware that the use of 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.	PASS
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
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	Thursday Jul 14 2022 21:54:08 GMT+0000 (Coordinated Universal Time)
Finished	Friday Jul 15 2022 06:46:24 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	JigsawToken.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	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 97

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
96   function add(uint256 a, uint256 b) internal pure returns (uint256) {
97   uint256 c = a + b;
98   require(c >= a, "SafeMath: addition overflow");
99
100   return c;
101
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 109

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
108   require(b <= a, errorMessage);
109   uint256 c = a - b;
110
111   return c;
112   }
113
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 122

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
121
122   uint256 c = a * b;
123   require(c / a == b, "SafeMath: multiplication overflow");
124
125   return c;
126
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 123

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
122   uint256 c = a * b;
123   require(c / a == b, "SafeMath: multiplication overflow");
124
125   return c;
126   }
127
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 134

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
133   require(b > 0, errorMessage);
134   uint256 c = a / b;
135   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
136
137   return c;
138
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 146

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
145   require(b != 0, errorMessage);
146   return a % b;
147   }
148   }
149
150
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 197

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
196   uint index = map.indexOf[key];
197   uint lastIndex = map.keys.length - 1;
198   address lastKey = map.keys[lastIndex];
199
200   map.indexOf[lastKey] = index;
201
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 383

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
382   // initialSupply
383   uint256 constant initialSupply = 1000000000 * (10**18);
384
385   // max wallet is 2.0% of initialSupply
386   uint256 public maxWalletAmount = initialSupply * 200 / 10000;
387
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 383

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
382   // initialSupply
383   uint256 constant initialSupply = 1000000000 * (10**18);
384
385   // max wallet is 2.0% of initialSupply
386   uint256 public maxWalletAmount = initialSupply * 200 / 10000;
387
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 386

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
385   // max wallet is 2.0% of initialSupply
386   uint256 public maxWalletAmount = initialSupply * 200 / 10000;
387
388   bool private _swapping;
389   uint256 public minimumTokensBeforeSwap = 25000000 * (10**18);
390
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 386

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
385   // max wallet is 2.0% of initialSupply
386   uint256 public maxWalletAmount = initialSupply * 200 / 10000;
387
388   bool private _swapping;
389   uint256 public minimumTokensBeforeSwap = 25000000 * (10**18);
390
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 389

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
388   bool private _swapping;
389   uint256 public minimumTokensBeforeSwap = 25000000 * (10**18);
390
391   address public liquidityWallet;
392   address public operationsWallet;
393
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 389

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
388   bool private _swapping;
389   uint256 public minimumTokensBeforeSwap = 25000000 * (10**18);
390
391   address public liquidityWallet;
392   address public operationsWallet;
393
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 502

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
501   require(!_isBlocked[account], "Jigsaw: Account is already blocked");
502   require((block.timestamp - _launchStartTimestamp) < _blockedTimeLimit, "Jigsaw:
Time to block accounts has expired");
503   _isBlocked[account] = true;
504   emit BlockedAccountChange(account, true);
505   }
506
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 605

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
604   if (!_isExcludedFromMaxWalletLimit[to]) {
605   require((balanceOf(to) + amount) <= maxWalletAmount, "Jigsaw: Expected wallet
amount exceeds the maxWalletAmount.");
606   }
607   }
608
609
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 630

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
629   if (takeFee && _totalFee > 0) {
630   uint256 fee = amount * _totalFee / 100;
631   amount = amount - fee;
632   super._transfer(from, address(this), fee);
633   }
634
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 630

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
629   if (takeFee && _totalFee > 0) {
630   uint256 fee = amount * _totalFee / 100;
631   amount = amount - fee;
632   super._transfer(from, address(this), fee);
633   }
634
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 631

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
630   uint256 fee = amount * _totalFee / 100;
631   amount = amount - fee;
632   super._transfer(from, address(this), fee);
633   }
634
635
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 646

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
645   if (isBuyFromLp) {
646   if (block.number - _launchBlockNumber <= 5) {
647   _liquidityFee = 100;
648   }
649   else {
650
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 665

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
664   }
665   _totalFee = _liquidityFee + _operationsFee + _jigsawFee;
666   emit FeesApplied(_liquidityFee, _operationsFee, _jigsawFee, _totalFee);
667   }
668   function _setBalance(address account, uint256 newBalance) private {
669
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 665

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
664   }
665   _totalFee = _liquidityFee + _operationsFee + _jigsawFee;
666   emit FeesApplied(_liquidityFee, _operationsFee, _jigsawFee, _totalFee);
667   }
668   function _setBalance(address account, uint256 newBalance) private {
669
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 717

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
716
717   uint256 amountToLiquify = contractBalance * _liquidityFee / _totalFee / 2;
718   uint256 amountToSwap = contractBalance - (amountToLiquify);
719
720   _swapTokensForETH(amountToSwap);
721
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 717

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
716
717   uint256 amountToLiquify = contractBalance * _liquidityFee / _totalFee / 2;
718   uint256 amountToSwap = contractBalance - (amountToLiquify);
719
720   _swapTokensForETH(amountToSwap);
721
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 717

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
716
717   uint256 amountToLiquify = contractBalance * _liquidityFee / _totalFee / 2;
718   uint256 amountToSwap = contractBalance - (amountToLiquify);
719
720   _swapTokensForETH(amountToSwap);
721
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 718

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
717   uint256 amountToLiquify = contractBalance * _liquidityFee / _totalFee / 2;
718   uint256 amountToSwap = contractBalance - (amountToLiquify);
719
720   _swapTokensForETH(amountToSwap);
721
722
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 722

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
721
722   uint256 ETHBalanceAfterSwap = address(this).balance - initialETHBalance;
723   uint256 totalETHFee = _totalFee - (_liquidityFee / 2);
724
725   uint256 amountETHLiquidity = ETHBalanceAfterSwap * _liquidityFee / totalETHFee / 2;
726
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 723

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
722   uint256 ETHBalanceAfterSwap = address(this).balance - initialETHBalance;
723   uint256 totalETHFee = _totalFee - (_liquidityFee / 2);
724
725   uint256 amountETHLiquidity = ETHBalanceAfterSwap * _liquidityFee / totalETHFee / 2;
726   uint256 amountETHOperations = ETHBalanceAfterSwap * _operationsFee / totalETHFee;
727
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 723

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
722   uint256 ETHBalanceAfterSwap = address(this).balance - initialETHBalance;
723   uint256 totalETHFee = _totalFee - (_liquidityFee / 2);
724
725   uint256 amountETHLiquidity = ETHBalanceAfterSwap * _liquidityFee / totalETHFee / 2;
726   uint256 amountETHOperations = ETHBalanceAfterSwap * _operationsFee / totalETHFee;
727
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 725

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
724
725   uint256 amountETHLiquidity = ETHBalanceAfterSwap * _liquidityFee / totalETHFee / 2;
726   uint256 amountETHOperations = ETHBalanceAfterSwap * _operationsFee / totalETHFee;
727   uint256 amountETHJigsaw = ETHBalanceAfterSwap - (amountETHLiquidity +
amountETHOperations);
728
729
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 725

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
724
725   uint256 amountETHLiquidity = ETHBalanceAfterSwap * _liquidityFee / totalETHFee / 2;
726   uint256 amountETHOperations = ETHBalanceAfterSwap * _operationsFee / totalETHFee;
727   uint256 amountETHJigsaw = ETHBalanceAfterSwap - (amountETHLiquidity +
amountETHOperations);
728
729
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 725

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
724
725   uint256 amountETHLiquidity = ETHBalanceAfterSwap * _liquidityFee / totalETHFee / 2;
726   uint256 amountETHOperations = ETHBalanceAfterSwap * _operationsFee / totalETHFee;
727   uint256 amountETHJigsaw = ETHBalanceAfterSwap - (amountETHLiquidity +
amountETHOperations);
728
729
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 726

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
725   uint256 amountETHLiquidity = ETHBalanceAfterSwap * _liquidityFee / totalETHFee / 2;
726   uint256 amountETHOperations = ETHBalanceAfterSwap * _operationsFee / totalETHFee;
727   uint256 amountETHJigsaw = ETHBalanceAfterSwap - (amountETHLiquidity +
amountETHOperations);
728
729   payable(operationsWallet).transfer(amountETHOperations);
730
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 726

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
725   uint256 amountETHLiquidity = ETHBalanceAfterSwap * _liquidityFee / totalETHFee / 2;
726   uint256 amountETHOperations = ETHBalanceAfterSwap * _operationsFee / totalETHFee;
727   uint256 amountETHJigsaw = ETHBalanceAfterSwap - (amountETHLiquidity +
amountETHOperations);
728
729   payable(operationsWallet).transfer(amountETHOperations);
730
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 727

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
726   uint256 amountETHOperations = ETHBalanceAfterSwap * _operationsFee / totalETHFee;
727   uint256 amountETHJigsaw = ETHBalanceAfterSwap - (amountETHLiquidity +
amountETHOperations);
728
729   payable(operationsWallet).transfer(amountETHOperations);
730   payable(jigsawWallet).transfer(amountETHJigsaw);
731
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 727

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
726   uint256 amountETHOperations = ETHBalanceAfterSwap * _operationsFee / totalETHFee;
727   uint256 amountETHJigsaw = ETHBalanceAfterSwap - (amountETHLiquidity +
amountETHOperations);
728
729   payable(operationsWallet).transfer(amountETHOperations);
730   payable(jigsawWallet).transfer(amountETHJigsaw);
731
```

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

LINE 197

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
196   uint index = map.indexOf[key];
197   uint lastIndex = map.keys.length - 1;
198   address lastKey = map.keys[lastIndex];
199
200   map.indexOf[lastKey] = index;
201
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 170

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
169   function getKeyAtIndex(Map storage map, uint index) public view returns (address) {
170   return map.keys[index];
171   }
172
173   function size(Map storage map) public view returns (uint) {
174
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 198

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
197   uint lastIndex = map.keys.length - 1;
198   address lastKey = map.keys[lastIndex];
199
200   map.indexOf[lastKey] = index;
201   delete map.indexOf[key];
202
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 203

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
202
203   map.keys[index] = lastKey;
204   map.keys.pop();
205   }
206   }
207
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 579

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
578   }
579   address account = tokenHoldersMap.keys[accountIndex];
580   return account;
581   }
582
583
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 741

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
740   address[] memory path = new address[](2);
741   path[0] = address(this);
742   path[1] = uniswapV2Router.WETH();
743   _approve(address(this), address(uniswapV2Router), tokenAmount);
744   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
745
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 742

low SEVERITY

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

Source File

- JigsawToken.sol

Locations

```
741   path[0] = address(this);
742   path[1] = uniswapV2Router.WETH();
743   _approve(address(this), address(uniswapV2Router), tokenAmount);
744   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
745   tokenAmount,
746
```

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

LINE 475

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

- JigsawToken.sol

Locations

```
474   _launchStartTimestamp = block.timestamp;
475   _launchBlockNumber = block.number;
476   }
477   }
478   function deactivateTrading() external onlyOwner {
479
```

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

LINE 646

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

- JigsawToken.sol

Locations

```
645   if (isBuyFromLp) {
646   if (block.number - _launchBlockNumber <= 5) {
647   _liquidityFee = 100;
648   }
649   else {
650
```

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.