Monokuma

Smart Contract Audit Report

SYSFIXED

27 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Monokuma	MONO	Binance Smart Chain

Addresses

Contract address	0xc0D0D9A7C2BbCB44BD5EBCf8954d7b54e6933E66
Contract deployer address	0xD57dBD3dA1E66410003934e50F3139f39fD86807

Project Website

https://monokumabsc.io/

Codebase

https://bscscan.com/address/0xc0D0D9A7C2BbCB44BD5EBCf8954d7b54e6933E66#code

SUMMARY

We present you the multi utility token which allow investors the greatest earning opportunity by burning tokens from the liquidity pool every 15 minutes to ensure the price of token is continuously rising. | No Private Sale | No Unlocked Tokens | No Team Tokens | Fair Tokenomics | Low Tax 3/3

Contract Summary

Documentation Quality

Monokuma 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 Monokuma 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 110, 160 and 169.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 123, 298, 326, 358, 396, 400, 401, 403, 404, 405, 499, 506, 507, 509, 510, 530, 531, 548, 565, 566, 585, 586, 606, 607, 608, 622, 624, 649, 651 and 655.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 6.
- SWC-110 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 534, 535, 607 and 608.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 456.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 582.

CONCLUSION

We have audited the Monokuma project released on January 2023 to discover issues and identify potential security vulnerabilities in Monokuma 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 code on Monokuma smart contract 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, weak sources of randomness, tx.origin as a part of authorization control 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.

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
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.	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.	ISSUE FOUND
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.	ISSUE FOUND
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	Thursday Jan 26 2023 13:07:19 GMT+0000 (Coordinated Universal Time)
Finished	Friday Jan 27 2023 05:07:32 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Monokuma.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-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	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-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 123

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
122   uint8 constant private _decimals = 18;
123   uint256 constant private _tTotal = startingSupply * 10**_decimals;
124
125   struct Fees {
126   uint16 buyFee;
127
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 298

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
297   if (_allowances[sender][msg.sender] != type(uint256).max) {
298   _allowances[sender][msg.sender] -= amount;
299   }
300
301   return _transfer(sender, recipient, amount);
302
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 326

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
325   if (timeSinceLastPair != 0) {
326   require(block.timestamp - timeSinceLastPair > 3 days, "3 Day cooldown.");
327   }
328   require(!lpPairs[pair], "Pair already added to list.");
329   lpPairs[pair] = true;
330
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 358

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
357   function getCirculatingSupply() public view returns (uint256) {
358   return (_tTotal - (balanceOf(DEAD) + balanceOf(address(0))));
359   }
360
361   //================================================ BLACKLIST
362
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 396

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
395   function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
returns (uint256) {
396   return((balanceOf(lpPair) * priceImpactInHundreds) / masterTaxDivisor);
397   }
398
399   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
400
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 400

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
399   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
400   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
401   swapAmount = (_tTotal * amountPercent) / amountDivisor;
402   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
403   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
404
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 401

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
400   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
401   swapAmount = (_tTotal * amountPercent) / amountDivisor;
402   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
403   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
404   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
405
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 403

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
402   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
403   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
404   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
405   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
406   }
407
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 404

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
403   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
404   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
405   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
406   }
407
408
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 405

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
404   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
405   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
406   }
407
408   function setPriceImpactSwapAmount(uint256 priceImpactSwapPercent) external
onlyOwner {
409
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 499

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
498   uint256 swapAmt = swapAmount;
499   if (piContractSwapsEnabled) { swapAmt = (balanceOf(lpPair) * piSwapPercent) /
masterTaxDivisor; }
500   if (contractTokenBalance >= swapAmt) { contractTokenBalance = swapAmt; }
501   contractSwap(contractTokenBalance);
502   }
503
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 506

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
505   if (lpBurnEnabled) {
506   if (block.timestamp - lpLastBurnStamp >= lpBurnTimeLimit) {
507   uint256 burnAmount = (_tOwned[lpPair] * lpBurnPercent) / masterTaxDivisor;
508   lpLastBurnStamp = block.timestamp;
509   _tOwned[lpPair] -= burnAmount;
510
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 507

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
506   if (block.timestamp - lpLastBurnStamp >= lpBurnTimeLimit) {
507   uint256 burnAmount = (_tOwned[lpPair] * lpBurnPercent) / masterTaxDivisor;
508   lpLastBurnStamp = block.timestamp;
509   _tOwned[lpPair] -= burnAmount;
510   _tOwned[DEAD] += burnAmount;
511
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 509

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
508   lpLastBurnStamp = block.timestamp;
509   _tOwned[lpPair] -= burnAmount;
510   _tOwned[DEAD] += burnAmount;
511   emit Transfer(lpPair, DEAD, burnAmount);
512   IV2Pair(lpPair).sync();
513
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 510

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
509   _tOwned[lpPair] -= burnAmount;
510   _tOwned[DEAD] += burnAmount;
511   emit Transfer(lpPair, DEAD, burnAmount);
512   IV2Pair(lpPair).sync();
513   }
514
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 530

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
529
530   uint256 toLiquify = ((contractTokenBalance * ratios.liquidity) / ratios.totalSwap)
/ 2;
531   uint256 swapAmt = contractTokenBalance - toLiquify;
532
533   address[] memory path = new address[](2);
534
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 531

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
530   uint256 toLiquify = ((contractTokenBalance * ratios.liquidity) / ratios.totalSwap)
/ 2;
531   uint256 swapAmt = contractTokenBalance - toLiquify;
532
533   address[] memory path = new address[](2);
534   path[0] = address(this);
535
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 548

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
547   uint256 amtBalance = address(this).balance;
548   uint256 liquidityBalance = (amtBalance * toLiquify) / swapAmt;
549
550   if (toLiquify > 0) {
551   try dexRouter.addLiquidityETH{value: liquidityBalance}(
552
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 565

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
564
565   amtBalance -= liquidityBalance;
566   ratios.totalSwap -= ratios.liquidity;
567   bool success;
568   (success,) = marketingWallet.call{value: address(this).balance, gas: 55000}("");
569
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 566

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
565   amtBalance -= liquidityBalance;
566   ratios.totalSwap -= ratios.liquidity;
567   bool success;
568   (success,) = marketingWallet.call{value: address(this).balance, gas: 55000}("");
569   }
570
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 585

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
584   allowedPresaleExclusion = false;
585   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
586   swapAmount = (balanceOf(lpPair) * 30) / 10000;
587   launchStamp = block.timestamp;
588   lpBurnEnabled = true;
589
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 586

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
585   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
586   swapAmount = (balanceOf(lpPair) * 30) / 10000;
587   launchStamp = block.timestamp;
588   lpBurnEnabled = true;
589   }
590
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 606

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
605   require(accounts.length == amounts.length, "Lengths do not match.");
606   for (uint16 i = 0; i < accounts.length; i++) {
607   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
608   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
609   }
610
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 607

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
606   for (uint16 i = 0; i < accounts.length; i++) {
607   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
608   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
609   }
610   }
611
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 608

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
607   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
608   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
609   }
610   }
611
612
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 622

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
621   }
622   _tOwned[from] -= amount;
623   uint256 amountReceived = (takeFee) ? takeTaxes(from, buy, sell, amount) : amount;
624   _tOwned[to] += amountReceived;
625   emit Transfer(from, to, amountReceived);
626
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 624

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
623   uint256 amountReceived = (takeFee) ? takeTaxes(from, buy, sell, amount) : amount;
624   _tOwned[to] += amountReceived;
625   emit Transfer(from, to, amountReceived);
626   if (!_hasLiqBeenAdded) {
627   _checkLiquidityAdd(from, to);
628
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 649

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
648   || block.chainid == 56)) { currentFee = 4500; }
649   uint256 feeAmount = amount * currentFee / masterTaxDivisor;
650   if (feeAmount > 0) {
651   _tOwned[address(this)] += feeAmount;
652   emit Transfer(from, address(this), feeAmount);
653
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 651

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
650   if (feeAmount > 0) {
651   _tOwned[address(this)] += feeAmount;
652   emit Transfer(from, address(this), feeAmount);
653   }
654
655
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 655

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
654
655   return amount - feeAmount;
656   }
657   }
658
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<0.9.0"". 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

- Monokuma.sol

Locations

```
5   // SPDX-License-Identifier: MIT
6   pragma solidity >=0.6.0 <0.9.0;
7
8   interface IERC20 {
9   function totalSupply() external view returns (uint256);
10
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 110

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
109   mapping (address => uint256) private _tOwned;
110   mapping (address => bool) lpPairs;
111   uint256 private timeSinceLastPair = 0;
112   mapping (address => mapping (address => uint256)) private _allowances;
113   mapping (address => bool) private _liquidityHolders;
114
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 160

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
159
160   bool inSwap;
161   bool public contractSwapEnabled = false;
162   uint256 public swapThreshold;
163   uint256 public swapAmount;
164
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 169

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
168   bool public _hasLiqBeenAdded = false;
169   Protections protections;
170   uint256 public launchStamp;
171
172   bool public lpBurnEnabled = false;
173
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 456

low SEVERITY

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

Source File

- Monokuma.sol

Locations

```
455   && to != _owner
456   && tx.origin != _owner
457   && !_liquidityHolders[to]
458   && !_liquidityHolders[from]
459   && to != DEAD
460
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 534

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
533   address[] memory path = new address[](2);
534   path[0] = address(this);
535   path[1] = dexRouter.WETH();
536
537   try dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
538
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 535

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
534   path[0] = address(this);
535   path[1] = dexRouter.WETH();
536
537   try dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
538   swapAmt,
539
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 607

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
606   for (uint16 i = 0; i < accounts.length; i++) {
607   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
608   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
609   }
610   }
611
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 608

low SEVERITY

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

Source File

- Monokuma.sol

Locations

```
607   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
608   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
609   }
610   }
611
612
```

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

LINE 582

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

- Monokuma.sol

Locations

```
581   emit ContractSwapEnabledUpdated(true);
582   try protections.setLaunch(lpPair, uint32(block.number), uint64(block.timestamp),
_decimals) {} catch {}
583   tradingEnabled = true;
584   allowedPresaleExclusion = false;
585   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
```

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.