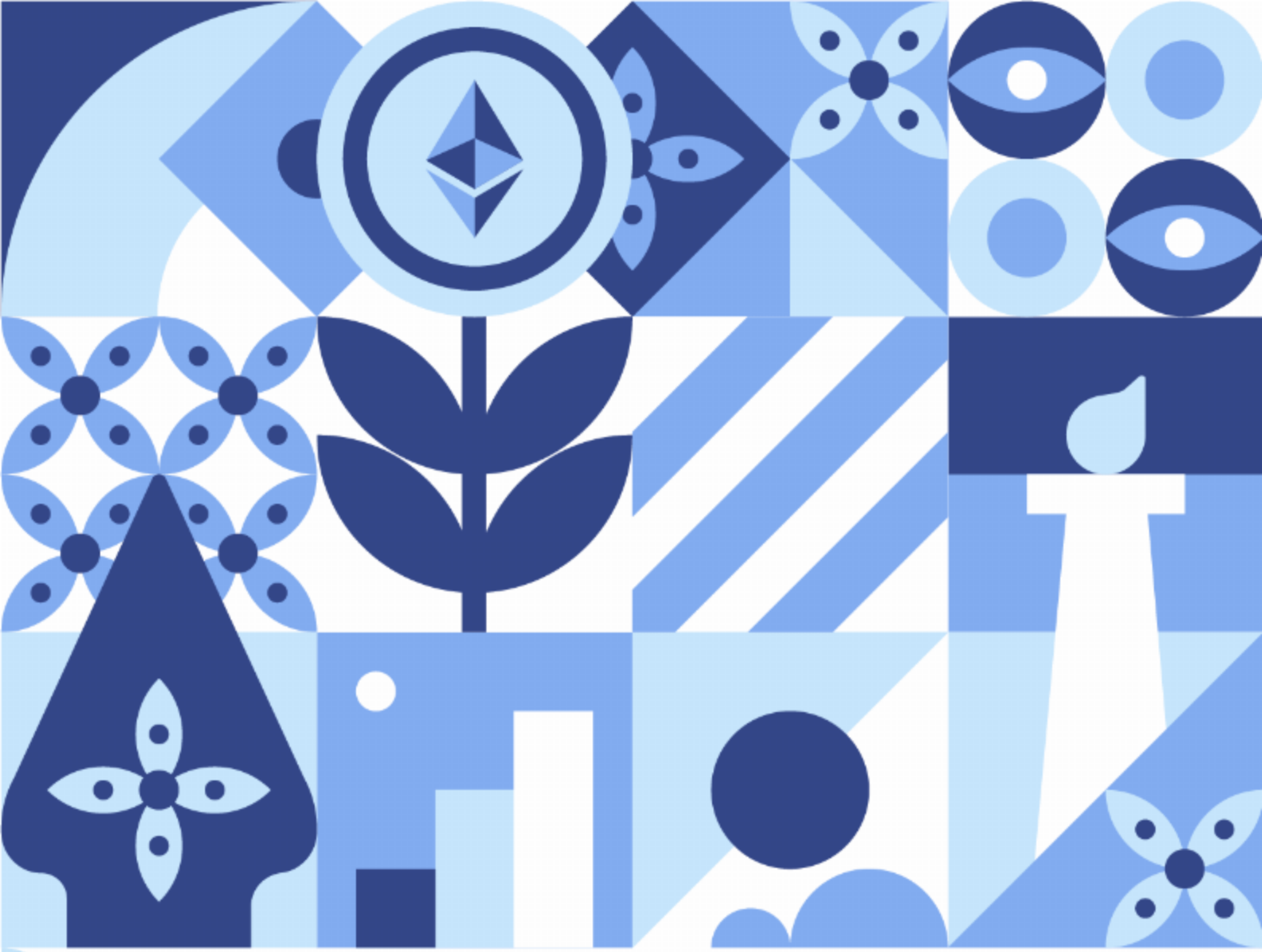

Zombie Inu

Smart Contract Audit Report

SYSFIXED

15 Oct 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
ZINU	ZINU	Ethereum

Addresses

Contract address	0xc5fdf3569af74f3b3e97e46a187a626352d2d508
Contract deployer address	0x885F5fd87E62eD2eBD0B0Bb1C295c4C43edEe5B5

Project Website

https://wearezinu.com/

Codebase

https://etherscan.io/address/0xc5fdf3569af74f3b3e97e46a187a626352d2d508#code

SUMMARY

ZINU is the ultimate, the definitive, the original Zombie. He knows how to mix with humanity as much as he knows how to fight the evil forces at play. Defiantly fearless, audacious to a fault, he's continually haunted by something darker.What that is, currently, we don't know, but partner we must, to help Zinu defeat it.

Contract Summary

Documentation Quality

ZINU 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 ZINU 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 68, 83, 97, 97, 111, 124, 138, 152, 164, 179, 194, 218, 241, 271, 587, 587, 816, 821, 846, 853, 853, 854, 854, 859, 941, 941, 941, 944, 944, 944, 947, 947, 949, 950, 953, 953, 955, 955, 955, 956, 956, 957, 957, 957, 957, 1011, 1018, 1093, 1099, 1099, 1099, 1101, 1101, 1101, 1101, 1115, 1116, 1117 and 1124.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 45, 278, 360, 388 and 465.
- 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 875, 876, 887, 887, 1011, 1019, 1116, 1116, 1124, 1124 and 1124.

CONCLUSION

We have audited the Zombie Inu project released on October 2022 to discover issues and identify potential security vulnerabilities in Zombie Inu 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 Zombie Inu 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, floating pragmas set on several lines 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.	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.	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.	PASS
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	Friday Oct 14 2022 22:08:55 GMT+0000 (Coordinated Universal Time)
Finished	Saturday Oct 15 2022 16:45:43 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	ZINU.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	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-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-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-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 68

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
67   /**
68   * @dev Returns the subtraction of two unsigned integers, with an overflow flag.
69   *
70   * _Available since v3.4._
71   */
72
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 83

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
82   * _Available since v3.4._
83   */
84   function tryMul(uint256 a, uint256 b) internal pure returns (bool, uint256) {
85   unchecked {
86   // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
87
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 97

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
96   /**
97   * @dev Returns the division of two unsigned integers, with a division by zero flag.
98   *
99   * _Available since v3.4._
100   */
101
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 97

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
96   /**
97   * @dev Returns the division of two unsigned integers, with a division by zero flag.
98   *
99   * _Available since v3.4._
100   */
101
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 111

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
110   *
111   * _Available since v3.4._
112   */
113   function tryMod(uint256 a, uint256 b) internal pure returns (bool, uint256) {
114   unchecked {
115
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 124

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
123   *
124   * Counterpart to Solidity's `+` operator.
125   *
126   * Requirements:
127   *
128
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 138

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
137   *
138   * Counterpart to Solidity's `-` operator.
139   *
140   * Requirements:
141   *
142
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 152

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
151   *
152   * Counterpart to Solidity's `*` operator.
153   *
154   * Requirements:
155   *
156
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 164

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
163   * @dev Returns the integer division of two unsigned integers, reverting on
164   * division by zero. The result is rounded towards zero.
165   *
166   * Counterpart to Solidity's `/` operator.
167   *
168
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 179

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
178   * reverting when dividing by zero.
179   *
180   * Counterpart to Solidity's `%` operator. This function uses a `revert`
181   * opcode (which leaves remaining gas untouched) while Solidity uses an
182   * invalid opcode to revert (consuming all remaining gas).
183
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 194

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
193   * @dev Returns the subtraction of two unsigned integers, reverting with custom
message on
194   * overflow (when the result is negative).
195   *
196   * CAUTION: This function is deprecated because it requires allocating memory for
the error
197   * message unnecessarily. For custom revert reasons use {trySub}.
198
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 218

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
217   * @dev Returns the integer division of two unsigned integers, reverting with custom
message on
218   * division by zero. The result is rounded towards zero.
219   *
220   * Counterpart to Solidity's `/` operator. Note: this function uses a
221   * `revert` opcode (which leaves remaining gas untouched) while Solidity
222
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 241

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
240   * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer
modulo),
241   * reverting with custom message when dividing by zero.
242   *
243   * CAUTION: This function is deprecated because it requires allocating memory for
the error
244   * message unnecessarily. For custom revert reasons use {tryMod}.
245
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 271

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
270
271   pragma solidity ^0.8.0;
272
273   /**
274   * @dev Interface of the ERC20 standard as defined in the EIP.
275
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 587

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
586   maxHodlAmount = _tTotal.mul(100).div(10000); //1%
587   contractSwapThreshold = _tTotal.mul(10).div(10000); //0.1%
588   buybackThreshold = 10; //10 wei
589
590   //Buy Fees
591
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 587

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
586   maxHodlAmount = _tTotal.mul(100).div(10000); //1%
587   contractSwapThreshold = _tTotal.mul(10).div(10000); //0.1%
588   buybackThreshold = 10; //10 wei
589
590   //Buy Fees
591
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 816

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
815   //Set for Sells
816   if (recipient == uniswapV2Pair && sender != address(uniswapV2Router)) {
817   sellTracker[sender] += amount;
818   }
819
820
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 821

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
820   // if the sell tracker equals or exceeds the amount of tokens bought,
821   // reset all variables here which resets the time-decaying sell tax logic.
822   if(sellTracker[sender] >= buyTracker[sender]) {
823   resetBuySellDecayTax(sender);
824   }
825
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 846

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
845   if(block.timestamp > getSellEarlyExpiration(_seller)) {
846   return 0;
847   }
848
849   uint256 _secondsAfterBuy = block.timestamp - lastBuyTimestamp[_seller];
850
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 853

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
852
853   function getSellEarlyExpiration(address _seller) private  view returns (uint256) {
854   return lastBuyTimestamp[_seller] == 0 ? 0 : lastBuyTimestamp[_seller] +
decayTaxExpiration;
855   }
856
857
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 853

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
852
853   function getSellEarlyExpiration(address _seller) private  view returns (uint256) {
854   return lastBuyTimestamp[_seller] == 0 ? 0 : lastBuyTimestamp[_seller] +
decayTaxExpiration;
855   }
856
857
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 854

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
853   function getSellEarlyExpiration(address _seller) private  view returns (uint256) {
854   return lastBuyTimestamp[_seller] == 0 ? 0 : lastBuyTimestamp[_seller] +
decayTaxExpiration;
855   }
856
857   function resetBuySellDecayTax(address _user) private {
858
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 854

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
853   function getSellEarlyExpiration(address _seller) private  view returns (uint256) {
854   return lastBuyTimestamp[_seller] == 0 ? 0 : lastBuyTimestamp[_seller] +
decayTaxExpiration;
855   }
856
857   function resetBuySellDecayTax(address _user) private {
858
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 859

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
858   buyTracker[_user] = balanceOf(_user);
859   lastBuyTimestamp[_user] = block.timestamp;
860   sellTracker[_user] = 0;
861   }
862
863
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 941

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
940   //Get tokens to stay in contract
941   uint tokensForLP = (tokens * sLPFee / totalTokensFee)/2; //alf of tokens goes to LP
and another half as ETH
942   uint tokensForBurn = (tokens * sBurnFee / totalTokensFee);
943
944   //Get tokens to swap for ETH
945
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 941

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
940   //Get tokens to stay in contract
941   uint tokensForLP = (tokens * sLPFee / totalTokensFee)/2; //alf of tokens goes to LP
and another half as ETH
942   uint tokensForBurn = (tokens * sBurnFee / totalTokensFee);
943
944   //Get tokens to swap for ETH
945
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 941

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
940   //Get tokens to stay in contract
941   uint tokensForLP = (tokens * sLPFee / totalTokensFee)/2; //alf of tokens goes to LP
and another half as ETH
942   uint tokensForBurn = (tokens * sBurnFee / totalTokensFee);
943
944   //Get tokens to swap for ETH
945
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 944

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
943
944   //Get tokens to swap for ETH
945   uint tokensForETHSwap = tokens - (tokensForBurn + tokensForBurn);
946
947   //Swap for eth
948
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 944

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
943
944   //Get tokens to swap for ETH
945   uint tokensForETHSwap = tokens - (tokensForBurn + tokensForBurn);
946
947   //Swap for eth
948
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 944

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
943
944   //Get tokens to swap for ETH
945   uint tokensForETHSwap = tokens - (tokensForBurn + tokensForBurn);
946
947   //Swap for eth
948
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 947

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
946
947   //Swap for eth
948   uint256 initialETHBalance = address(this).balance;
949   swapTokensForEth(tokensForETHSwap);
950   uint256 newETHBalance = address(this).balance.sub(initialETHBalance);
951
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 947

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
946
947   //Swap for eth
948   uint256 initialETHBalance = address(this).balance;
949   swapTokensForEth(tokensForETHSwap);
950   uint256 newETHBalance = address(this).balance.sub(initialETHBalance);
951
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 949

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
948   uint256 initialETHBalance = address(this).balance;
949   swapTokensForEth(tokensForETHSwap);
950   uint256 newETHBalance = address(this).balance.sub(initialETHBalance);
951
952   uint256 ethForMarketing = newETHBalance * sMarketingFee / (totalTokensFee -
(sLPFee/2) - sBurnFee);
953
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 950

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
949   swapTokensForEth(tokensForETHSwap);
950   uint256 newETHBalance = address(this).balance.sub(initialETHBalance);
951
952   uint256 ethForMarketing = newETHBalance * sMarketingFee / (totalTokensFee -
(sLPFee/2) - sBurnFee);
953   uint256 ethForLP = newETHBalance * (sLPFee/2) / (totalTokensFee - (sLPFee/2) -
sBurnFee);
954
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 953

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
952   uint256 ethForMarketing = newETHBalance * sMarketingFee / (totalTokensFee -
(sLPFee/2) - sBurnFee);
953   uint256 ethForLP = newETHBalance * (sLPFee/2) / (totalTokensFee - (sLPFee/2) -
sBurnFee);
954
955   //Send eth share to distribute to tax wallets
956   sendETHToFee(ethForMarketing);
957
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 953

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
952   uint256 ethForMarketing = newETHBalance * sMarketingFee / (totalTokensFee -
(sLPFee/2) - sBurnFee);
953   uint256 ethForLP = newETHBalance * (sLPFee/2) / (totalTokensFee - (sLPFee/2) -
sBurnFee);
954
955   //Send eth share to distribute to tax wallets
956   sendETHToFee(ethForMarketing);
957
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 955

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
954
955   //Send eth share to distribute to tax wallets
956   sendETHToFee(ethForMarketing);
957   //Send lp share along with tokens to add LP
958   addLiquidity(tokensForLP, ethForLP);
959
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 955

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
954
955   //Send eth share to distribute to tax wallets
956   sendETHToFee(ethForMarketing);
957   //Send lp share along with tokens to add LP
958   addLiquidity(tokensForLP, ethForLP);
959
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 955

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
954
955   //Send eth share to distribute to tax wallets
956   sendETHToFee(ethForMarketing);
957   //Send lp share along with tokens to add LP
958   addLiquidity(tokensForLP, ethForLP);
959
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 956

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
955   //Send eth share to distribute to tax wallets
956   sendETHToFee(ethForMarketing);
957   //Send lp share along with tokens to add LP
958   addLiquidity(tokensForLP, ethForLP);
959   //Burn
960
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 956

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
955   //Send eth share to distribute to tax wallets
956   sendETHToFee(ethForMarketing);
957   //Send lp share along with tokens to add LP
958   addLiquidity(tokensForLP, ethForLP);
959   //Burn
960
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 957

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
956   sendETHToFee(ethForMarketing);
957   //Send lp share along with tokens to add LP
958   addLiquidity(tokensForLP, ethForLP);
959   //Burn
960   _burn(address(this), tokensForBurn);
961
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 957

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
956   sendETHToFee(ethForMarketing);
957   //Send lp share along with tokens to add LP
958   addLiquidity(tokensForLP, ethForLP);
959   //Burn
960   _burn(address(this), tokensForBurn);
961
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 957

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
956   sendETHToFee(ethForMarketing);
957   //Send lp share along with tokens to add LP
958   addLiquidity(tokensForLP, ethForLP);
959   //Burn
960   _burn(address(this), tokensForBurn);
961
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 957

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
956   sendETHToFee(ethForMarketing);
957   //Send lp share along with tokens to add LP
958   addLiquidity(tokensForLP, ethForLP);
959   //Burn
960   _burn(address(this), tokensForBurn);
961
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1011

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1010   /// @notice Block address from transfer
1011   function blockMultipleBots(address[] calldata _bots, bool status) public onlyOwner
{
1012   for(uint256 i = 0; i < _bots.length; i++) {
1013   bots[_bots[i]] = status;
1014   }
1015
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1018

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1017   /// @notice Enable disable trading
1018   function setTrading(bool _tradingOpen) public onlyOwner {
1019   tradingOpen = _tradingOpen;
1020   }
1021
1022
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1093

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1092
1093   function readFees() external view returns (uint _totalBuyFee, uint _totalSellFee,
uint _burnFeeBuy, uint _burnFeeSell, uint _marketingFeeBuy, uint _marketingFeeSell, uint
_liquidityFeeBuy, uint _liquidityFeeSell, uint _buybackFeeBuy, uint _buybackFeeSell, uint
maxEarlySellFee) {
1094   return (
1095   bBurnFee+bMarketingFee+bLPFee+bBuybackFee,
1096   sBurnFee+sMarketingFee+sLPFee+sBuybackFee+sEarlySellFee,
1097
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1099

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1098   sBurnFee,
1099   bMarketingFee,
1100   sMarketingFee,
1101   bLPFee,
1102   sLPFee,
1103
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1099

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1098   sBurnFee,
1099   bMarketingFee,
1100   sMarketingFee,
1101   bLPFee,
1102   sLPFee,
1103
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1099

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1098   sBurnFee,
1099   bMarketingFee,
1100   sMarketingFee,
1101   bLPFee,
1102   sLPFee,
1103
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1101

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1100   sMarketingFee,
1101   bLPFee,
1102   sLPFee,
1103   bBuybackFee,
1104   sBuybackFee,
1105
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1101

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1100   sMarketingFee,
1101   bLPFee,
1102   sLPFee,
1103   bBuybackFee,
1104   sBuybackFee,
1105
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1101

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1100   sMarketingFee,
1101   bLPFee,
1102   sLPFee,
1103   bBuybackFee,
1104   sBuybackFee,
1105
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1101

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1100   sMarketingFee,
1101   bLPFee,
1102   sLPFee,
1103   bBuybackFee,
1104   sBuybackFee,
1105
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1115

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1114
1115   //Suppose to airdrop holders who bought long back and don't want to reset their
decaytax
1116   if(overrideTracker) {
1117   //Override buytracker
1118   buyTracker[addresses[i]] += amounts[i];
1119
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1116

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1115   //Suppose to airdrop holders who bought long back and don't want to reset their
decaytax
1116   if(overrideTracker) {
1117   //Override buytracker
1118   buyTracker[addresses[i]] += amounts[i];
1119   lastBuyTimestamp[addresses[i]] = trackerTimestamp;
1120
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1117

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1116   if(overrideTracker) {
1117   //Override buytracker
1118   buyTracker[addresses[i]] += amounts[i];
1119   lastBuyTimestamp[addresses[i]] = trackerTimestamp;
1120   }
1121
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1124

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1123
1124    }
1125
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 45

low SEVERITY

The current pragma Solidity directive is ""^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

- ZINU.sol

Locations

```
44   // This version of SafeMath should only be used with Solidity 0.8 or later,
45   // because it relies on the compiler's built in overflow checks.
46
47   /**
48   * @dev Wrappers over Solidity's arithmetic operations.
49
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 278

low SEVERITY

The current pragma Solidity directive is ""^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

- ZINU.sol

Locations

```
277   /**
278   * @dev Emitted when `value` tokens are moved from one account (`from`) to
279   * another (`to`).
280   *
281   * Note that `value` may be zero.
282
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 360

low SEVERITY

The current pragma Solidity directive is ""^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

- ZINU.sol

Locations

```
359   * @dev Provides information about the current execution context, including the
360   * sender of the transaction and its data. While these are generally available
361   * via msg.sender and msg.data, they should not be accessed in such a direct
362   * manner, since when dealing with meta-transactions the account sending and
363   * paying for execution may not be the actual sender (as far as an application
364
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 388

low SEVERITY

The current pragma Solidity directive is ""^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

- ZINU.sol

Locations

```
387   * @dev Contract module which provides a basic access control mechanism, where
388   * there is an account (an owner) that can be granted exclusive access to
389   * specific functions.
390   *
391   * By default, the owner account will be the one that deploys the contract. This
392
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 465

low SEVERITY

The current pragma Solidity directive is ""^0.8.7"". 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

- ZINU.sol

Locations

```
464
465   interface IUniswapV2Router02 {
466
467   function swapExactETHForTokensSupportingFeeOnTransferTokens(
468   uint256 amountOutMin,
469
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 875

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
874   uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(
875   0, // accept any amount of Tokens
876   path,
877   _buybackTokenReceiver, //Send bought tokens to this address
878   block.timestamp.add(300)
879
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 876

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
875   0, // accept any amount of Tokens
876   path,
877   _buybackTokenReceiver, //Send bought tokens to this address
878   block.timestamp.add(300)
879   );
880
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 887

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
886   _approve(address(this), address(uniswapV2Router), tokenAmount);
887   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
888   tokenAmount,
889   0,
890   path,
891
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 887

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
886   _approve(address(this), address(uniswapV2Router), tokenAmount);
887   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
888   tokenAmount,
889   0,
890   path,
891
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1011

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1010   /// @notice Block address from transfer
1011   function blockMultipleBots(address[] calldata _bots, bool status) public onlyOwner
{
1012   for(uint256 i = 0; i < _bots.length; i++) {
1013   bots[_bots[i]] = status;
1014   }
1015
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1019

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1018   function setTrading(bool _tradingOpen) public onlyOwner {
1019   tradingOpen = _tradingOpen;
1020   }
1021
1022   /// @notice Enable/Disable contract fee distribution
1023
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1116

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1115   //Suppose to airdrop holders who bought long back and don't want to reset their
decaytax
1116   if(overrideTracker) {
1117   //Override buytracker
1118   buyTracker[addresses[i]] += amounts[i];
1119   lastBuyTimestamp[addresses[i]] = trackerTimestamp;
1120
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1116

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1115   //Suppose to airdrop holders who bought long back and don't want to reset their
decaytax
1116   if(overrideTracker) {
1117   //Override buytracker
1118   buyTracker[addresses[i]] += amounts[i];
1119   lastBuyTimestamp[addresses[i]] = trackerTimestamp;
1120
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1124

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1123
1124    }
1125
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1124

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1123
1124    }
1125
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1124

low SEVERITY

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

Source File

- ZINU.sol

Locations

```
1123
1124    }
1125
```

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.