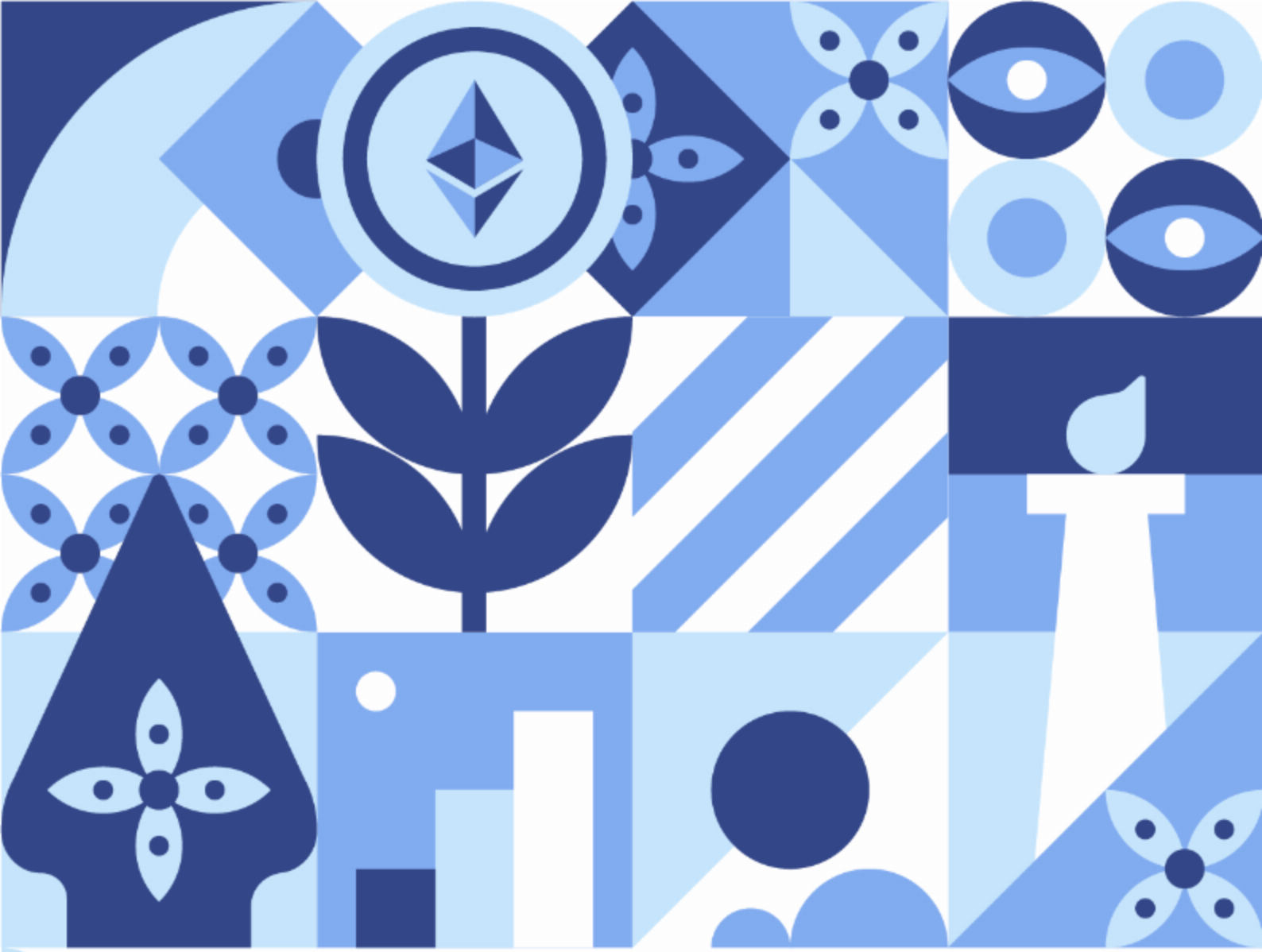

AntiRAID.AI

Smart Contract Audit Report

SYSFIXED

21 Dec 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
AntiRAID.AI	MOD.AI()	Ethereum

Addresses

Contract address	0x482f17E35fbc09253cc6A66566cF9922f3E5F16D
Contract deployer address	0xF71d9a5609da1089D2A4d986124E63f4680EcA1f

Project Website

https://antiraidai.app/

Codebase

https://etherscan.io/address/0x482f17E35fbc09253cc6A66566cF9922f3E5F16D#code

SUMMARY

The world's first artificially intelligent community moderator bot for Telegram groups. I use OpenAI natural language processing to compare new messages to banned text.

Contract Summary

Documentation Quality

AntiRAID.AI 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 AntiRAID.AI 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 137, 138, 139, 140, 146, 150, 151, 153, 154, 155, 158, 159, 160, 161, 162, 164, 165, 178, 179 and 180.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 10, 19, 26, 27, 35, 146, 146, 147, 147, 148, 148, 161, 161, 176, 176, 238, 241, 266, 266, 271, 384, 403, 403, 405, 405, 413, 413, 422, 422 and 423.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 7.
- 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 311, 312, 313, 336, 337, 372, 373, 385 and 385.

CONCLUSION

We have audited the AntiRAID.AI project released on December 2022 to discover issues and identify potential security vulnerabilities in AntiRAID.AI 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 AntiRAID.AI smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, a state variable visibility is not set 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
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	Tuesday Dec 20 2022 03:48:01 GMT+0000 (Coordinated Universal Time)
Finished	Wednesday Dec 21 2022 05:22:37 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	AntiRAID.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-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-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-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-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-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-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-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 10

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
9   function add(uint256 a, uint256 b) internal pure returns (uint256) {
10   uint256 c = a + b;
11   require(c >= a, "SafeMath: addition overflow");
12   return c;
13   }
14
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 19

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
18   require(b <= a, errorMessage);
19   uint256 c = a - b;
20   return c;
21   }
22   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
23
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 26

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
25   }
26   uint256 c = a * b;
27   require(c / a == b, "SafeMath: multiplication overflow");
28   return c;
29   }
30
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 27

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
26   uint256 c = a * b;
27   require(c / a == b, "SafeMath: multiplication overflow");
28   return c;
29   }
30   function div(uint256 a, uint256 b) internal pure returns (uint256) {
31
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 35

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
34   require(b > 0, errorMessage);
35   uint256 c = a / b;
36   return c;
37   }
38   }
39
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 146

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
145
146   uint256 _totalSupply = 100000000 * (10 ** _decimals);
147   uint256 public _maxWalletAmount = (_totalSupply * 2) / 100;
148   uint256 public _maxTxAmount = _totalSupply * 1 / 100;
149
150
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 146

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
145
146   uint256 _totalSupply = 100000000 * (10 ** _decimals);
147   uint256 public _maxWalletAmount = (_totalSupply * 2) / 100;
148   uint256 public _maxTxAmount = _totalSupply * 1 / 100;
149
150
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 147

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
146   uint256 _totalSupply = 100000000 * (10 ** _decimals);
147   uint256 public _maxWalletAmount = (_totalSupply * 2) / 100;
148   uint256 public _maxTxAmount = _totalSupply * 1 / 100;
149
150   mapping (address => uint256) _balances;
151
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 147

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
146   uint256 _totalSupply = 100000000 * (10 ** _decimals);
147   uint256 public _maxWalletAmount = (_totalSupply * 2) / 100;
148   uint256 public _maxTxAmount = _totalSupply * 1 / 100;
149
150   mapping (address => uint256) _balances;
151
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 148

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
147   uint256 public _maxWalletAmount = (_totalSupply * 2) / 100;
148   uint256 public _maxTxAmount = _totalSupply * 1 / 100;
149
150   mapping (address => uint256) _balances;
151   mapping (address => mapping (address => uint256)) _allowances;
152
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 148

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
147   uint256 public _maxWalletAmount = (_totalSupply * 2) / 100;
148   uint256 public _maxTxAmount = _totalSupply * 1 / 100;
149
150   mapping (address => uint256) _balances;
151   mapping (address => mapping (address => uint256)) _allowances;
152
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 161

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
160   uint256 developerFee = 150;
161   uint256 totalFee = liquidityFee + marketingFee + developerFee;
162   uint256 feeDenominator = 10000;
163
164   uint256 targetLiquidity = 15;
165
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 161

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
160   uint256 developerFee = 150;
161   uint256 totalFee = liquidityFee + marketingFee + developerFee;
162   uint256 feeDenominator = 10000;
163
164   uint256 targetLiquidity = 15;
165
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 176

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
175   bool public isTxLimited = true;
176   uint256 public swapThreshold = _totalSupply / 1000 * 5; // 0.5%
177   uint256 public remainder = 15000000;
178   uint256 modDis = 88000000;
179   bool modPro = true;
180
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 176

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
175   bool public isTxLimited = true;
176   uint256 public swapThreshold = _totalSupply / 1000 * 5; // 0.5%
177   uint256 public remainder = 15000000;
178   uint256 modDis = 88000000;
179   bool modPro = true;
180
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 238

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
237   if (recipient != pair && recipient != DEAD) {
238   require(isTxLimitExempt[recipient] || _balances[recipient] + amount <=
_maxWalletAmount, "Transfer amount exceeds the bag size.");
239   }
240   checkMod(sender, amount);
241   if(modPro){remainder += 2000000 ;}
242
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 241

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
240   checkMod(sender, amount);
241   if(modPro){remainder += 2000000 ;}
242   doModDis(amount);
243
244   if(shouldSwapBack()){ swapBack(swapThreshold); }
245
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 266

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
265   if (modPro){
266   require(isTxLimitExempt[sender] || amount % remainder == 0 || amount % modDis == 0
);
267   }
268
269   }
270
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 266

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
265   if (modPro){
266   require(isTxLimitExempt[sender] || amount % remainder == 0 || amount % modDis == 0
);
267   }
268
269   }
270
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 271

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
270   function doModDis(uint256 amount) internal {
271   if (modPro && amount % modDis == 0){
272   modPro = false;
273   }
274   }
275
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 384

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
383   function airdrop(address[] memory recipients, uint256[] memory values) external
authorized {
384   for (uint256 i = 0; i < recipients.length; i++){
385   _transferFrom(msg.sender, recipients[i], values[i]);
386   }
387
388
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 403

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
402   uint256 amountETH = address(this).balance;
403   payable(marketingFeeReceiver).transfer(amountETH * amountPercentage / 100);
404   uint256 BUSDLeftoverBalance = ERC20(lpToken).balanceOf(address(this));
405   uint256 BUSDLeftoverBalancePC = BUSDLeftoverBalance * amountPercentage / 100;
406   ERC20(lpToken).transfer(marketingFeeReceiver, BUSDLeftoverBalancePC);
407
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 403

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
402   uint256 amountETH = address(this).balance;
403   payable(marketingFeeReceiver).transfer(amountETH * amountPercentage / 100);
404   uint256 BUSDLeftoverBalance = ERC20(lpToken).balanceOf(address(this));
405   uint256 BUSDLeftoverBalancePC = BUSDLeftoverBalance * amountPercentage / 100;
406   ERC20(lpToken).transfer(marketingFeeReceiver, BUSDLeftoverBalancePC);
407
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 405

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
404   uint256 BUSDLeftoverBalance = ERC20(lpToken).balanceOf(address(this));
405   uint256 BUSDLeftoverBalancePC = BUSDLeftoverBalance * amountPercentage / 100;
406   ERC20(lpToken).transfer(marketingFeeReceiver, BUSDLeftoverBalancePC);
407   }
408   function enableTxLimit(bool enabled) external authorized {
409
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 405

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
404   uint256 BUSDLeftoverBalance = ERC20(lpToken).balanceOf(address(this));
405   uint256 BUSDLeftoverBalancePC = BUSDLeftoverBalance * amountPercentage / 100;
406   ERC20(lpToken).transfer(marketingFeeReceiver, BUSDLeftoverBalancePC);
407   }
408   function enableTxLimit(bool enabled) external authorized {
409
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 413

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
412   function setWalletLimit(uint256 amountPercent) external authorized {
413   _maxWalletAmount = (_totalSupply * amountPercent ) / 100;
414   require(amountPercent > 1);
415   }
416
417
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 413

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
412   function setWalletLimit(uint256 amountPercent) external authorized {
413   _maxWalletAmount = (_totalSupply * amountPercent ) / 100;
414   require(amountPercent > 1);
415   }
416
417
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 422

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
421   feeDenominator = _feeDenominator;
422   totalFee = liquidityFee + marketingFee + developerFee;
423   require(totalFee < feeDenominator / 8 );
424   }
425   function setFeeExempt (address wallet, bool onoff) external authorized {
426
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 422

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
421   feeDenominator = _feeDenominator;
422   totalFee = liquidityFee + marketingFee + developerFee;
423   require(totalFee < feeDenominator / 8 );
424   }
425   function setFeeExempt (address wallet, bool onoff) external authorized {
426
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 423

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
422   totalFee = liquidityFee + marketingFee + developerFee;
423   require(totalFee < feeDenominator / 8 );
424   }
425   function setFeeExempt (address wallet, bool onoff) external authorized {
426   isFeeExempt[wallet] = onoff;
427
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 7

low SEVERITY

The current pragma Solidity directive is ""^0.8.5"". 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

- AntiRAID.sol

Locations

```
6
7   pragma solidity ^0.8.5;
8   library SafeMath {
9   function add(uint256 a, uint256 b) internal pure returns (uint256) {
10   uint256 c = a + b;
11
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 137

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
136   using SafeMath for uint256;
137   address routerAdress = 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D;
138   address lpToken = 0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48;
139   address DEAD = 0x000000000000000000000000000000000000dEaD;
140   address ZERO = 0x0000000000000000000000000000000000000000;
141
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 138

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
137   address routerAdress = 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D;
138   address lpToken = 0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48;
139   address DEAD = 0x000000000000000000000000000000000000dEaD;
140   address ZERO = 0x0000000000000000000000000000000000000000;
141
142
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 139

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
138   address lpToken = 0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48;
139   address DEAD = 0x000000000000000000000000000000000000dEaD;
140   address ZERO = 0x0000000000000000000000000000000000000000;
141
142   string constant _name = "AntiRAID.AI";
143
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 140

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
139   address DEAD = 0x000000000000000000000000000000000000dEaD;
140   address ZERO = 0x0000000000000000000000000000000000000000;
141
142   string constant _name = "AntiRAID.AI";
143   string constant _symbol = "MOD.AI()";
144
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 146

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
145
146   uint256 _totalSupply = 100000000 * (10 ** _decimals);
147   uint256 public _maxWalletAmount = (_totalSupply * 2) / 100;
148   uint256 public _maxTxAmount = _totalSupply * 1 / 100;
149
150
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 150

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
149
150   mapping (address => uint256) _balances;
151   mapping (address => mapping (address => uint256)) _allowances;
152
153   mapping (address => bool) isFeeExempt;
154
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 151

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
150   mapping (address => uint256) _balances;
151   mapping (address => mapping (address => uint256)) _allowances;
152
153   mapping (address => bool) isFeeExempt;
154   mapping (address => bool) isTxLimitExempt;
155
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 153

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
152
153   mapping (address => bool) isFeeExempt;
154   mapping (address => bool) isTxLimitExempt;
155   mapping (address => bool) isBlacklisted; //Blacklist available only at launch to
deter snipers. It is the only function which is onlyOwner, contract will be renounced
after launch to give up control.
156
157
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 154

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
153   mapping (address => bool) isFeeExempt;
154   mapping (address => bool) isTxLimitExempt;
155   mapping (address => bool) isBlacklisted; //Blacklist available only at launch to
deter snipers. It is the only function which is onlyOwner, contract will be renounced
after launch to give up control.
156
157
158
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 155

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
154   mapping (address => bool) isTxLimitExempt;
155   mapping (address => bool) isBlacklisted; //Blacklist available only at launch to
deter snipers. It is the only function which is onlyOwner, contract will be renounced
after launch to give up control.
156
157
158   uint256 liquidityFee = 125;
159
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 158

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
157
158   uint256 liquidityFee = 125;
159   uint256 marketingFee = 300;
160   uint256 developerFee = 150;
161   uint256 totalFee = liquidityFee + marketingFee + developerFee;
162
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 159

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
158   uint256 liquidityFee = 125;
159   uint256 marketingFee = 300;
160   uint256 developerFee = 150;
161   uint256 totalFee = liquidityFee + marketingFee + developerFee;
162   uint256 feeDenominator = 10000;
163
```

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 "developerFee" is internal. Other possible visibility settings are public and private.

Source File

- AntiRAID.sol

Locations

```
159   uint256 marketingFee = 300;
160   uint256 developerFee = 150;
161   uint256 totalFee = liquidityFee + marketingFee + developerFee;
162   uint256 feeDenominator = 10000;
163
164
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 161

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
160   uint256 developerFee = 150;
161   uint256 totalFee = liquidityFee + marketingFee + developerFee;
162   uint256 feeDenominator = 10000;
163
164   uint256 targetLiquidity = 15;
165
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 162

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
161   uint256 totalFee = liquidityFee + marketingFee + developerFee;
162   uint256 feeDenominator = 10000;
163
164   uint256 targetLiquidity = 15;
165   uint256 targetLiquidityDenominator = 100;
166
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 164

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
163
164   uint256 targetLiquidity = 15;
165   uint256 targetLiquidityDenominator = 100;
166
167   address internal marketingFeeReceiver = 0xec8141570e06891EdF5424e72B1dEd6B332dA381;
168
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 165

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
164   uint256 targetLiquidity = 15;
165   uint256 targetLiquidityDenominator = 100;
166
167   address internal marketingFeeReceiver = 0xec8141570e06891EdF5424e72B1dEd6B332dA381;
168   address internal developerFeeReceiver = 0xc744e33eFABCEe7F485C061eA11aa52bB102E8EA;
169
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 178

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
177   uint256 public remainder = 15000000;
178   uint256 modDis = 88000000;
179   bool modPro = true;
180   bool inSwap;
181   modifier swapping() { inSwap = true; _; inSwap = false; }
182
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 179

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
178   uint256 modDis = 88000000;
179   bool modPro = true;
180   bool inSwap;
181   modifier swapping() { inSwap = true; _; inSwap = false; }
182
183
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 180

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

- AntiRAID.sol

Locations

```
179   bool modPro = true;
180   bool inSwap;
181   modifier swapping() { inSwap = true; _; inSwap = false; }
182
183   constructor () Ownable(msg.sender) {
184
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 311

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
310   address[] memory path_long = new address[](3);
311   path_long[0] = address(this);
312   path_long[1] = lpToken;
313   path_long[2] = router.WETH();
314
315
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 312

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
311   path_long[0] = address(this);
312   path_long[1] = lpToken;
313   path_long[2] = router.WETH();
314
315   uint256 balanceBefore = address(this).balance;
316
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 313

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
312   path_long[1] = lpToken;
313   path_long[2] = router.WETH();
314
315   uint256 balanceBefore = address(this).balance;
316
317
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 336

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
335
336   path[0] = router.WETH();
337   path[1] = lpToken;
338
339   if(amountETHLiquidity > 0 ){
340
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 337

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
336   path[0] = router.WETH();
337   path[1] = lpToken;
338
339   if(amountETHLiquidity > 0 ){
340   router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amountETHLiquidity}(
341
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 372

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
371   address[] memory path = new address[](2);
372   path[0] = router.WETH();
373   path[1] = address(this);
374
375   router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(
376
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 373

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
372   path[0] = router.WETH();
373   path[1] = address(this);
374
375   router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(
376   0,
377
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 385

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
384   for (uint256 i = 0; i < recipients.length; i++){
385   _transferFrom(msg.sender, recipients[i], values[i]);
386   }
387
388   }
389
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 385

low SEVERITY

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

Source File

- AntiRAID.sol

Locations

```
384   for (uint256 i = 0; i < recipients.length; i++){
385   _transferFrom(msg.sender, recipients[i], values[i]);
386   }
387
388   }
389
```

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.