RabbitAI

Smart Contract Audit Report

SYSFIXED

29 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
RabbitAI	Rabbit Ye...	Binance Smart Chain

Addresses

Contract address	0xfC64BB63E5a91250114d12fF0b9376b125660656
Contract deployer address	0xeeeB10e1462873850d88A2c335215E755E3aF157

Project Website

https://www.rabbityear2023.net/

Codebase

https://bscscan.com/address/0xfC64BB63E5a91250114d12fF0b9376b125660656#code

SUMMARY

RabbitAI is a revolutionary new platform that brings the power of AI and blockchain technology together and is operated as a DAO. With our AI-assisted dashboard, users can easily create and deploy complex blockchain projects in minutes, without the need for expensive development costs or long waiting times. Plus, with our unique token burn model, every time a user deploys a contract on our platform, they are contributing to the stability and growth of the Rabbit token. Join the RabbitAI revolution! 1% tax.

Contract Summary

Documentation Quality

RabbitAI 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 RabbitAI 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, 149 and 157.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 123, 123, 305, 333, 365, 365, 411, 423, 423, 427, 427, 428, 428, 430, 430, 431, 432, 512, 512, 568, 568, 569, 569, 586, 587, 587, 588, 588, 602, 604, 628, 628, 630 and 634.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 6.
- 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 528, 529, 587, 588 and 588.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 469.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 565.

CONCLUSION

We have audited the RabbitAI project released on January 2023 to discover issues and identify potential security vulnerabilities in RabbitAI 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 RabbitAI 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, state variable visibility is not set, and the Potential use of "block.number" as a source of randomness. We recommended specifying 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, and 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.

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.	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
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	ISSUE FOUND
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

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

SMART CONTRACT ANALYSIS

Started	Saturday Jan 28 2023 10:30:55 GMT+0000 (Coordinated Universal Time)
Finished	Sunday Jan 29 2023 22:54:50 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	RabbitAI.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-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-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

- RabbitAI.sol

Locations

```
122   uint8 constant private _decimals = 9;
123   uint256 constant private _tTotal = startingSupply * 10**_decimals;
124
125   struct Fees {
126   uint16 buyFee;
127
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 123

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
122   uint8 constant private _decimals = 9;
123   uint256 constant private _tTotal = startingSupply * 10**_decimals;
124
125   struct Fees {
126   uint16 buyFee;
127
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 305

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
304   if (_allowances[sender][msg.sender] != type(uint256).max) {
305   _allowances[sender][msg.sender] -= amount;
306   }
307
308   return _transfer(sender, recipient, amount);
309
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 333

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
332   if (timeSinceLastPair != 0) {
333   require(block.timestamp - timeSinceLastPair > 3 days, "3 Day cooldown.");
334   }
335   require(!lpPairs[pair], "Pair already added to list.");
336   lpPairs[pair] = true;
337
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 365

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
364   function getCirculatingSupply() public view returns (uint256) {
365   return (_tTotal - (balanceOf(DEAD) + balanceOf(address(0))));
366   }
367
368
369
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 365

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
364   function getCirculatingSupply() public view returns (uint256) {
365   return (_tTotal - (balanceOf(DEAD) + balanceOf(address(0))));
366   }
367
368
369
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 411

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
410   "Cannot exceed maximums.");
411   require(buyFee + sellFee <= maxRoundtripTax, "Cannot exceed roundtrip maximum.");
412   _taxRates.buyFee = buyFee;
413   _taxRates.sellFee = sellFee;
414   _taxRates.transferFee = transferFee;
415
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 423

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
422   function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
returns (uint256) {
423   return((balanceOf(lpPair) * priceImpactInHundreds) / masterTaxDivisor);
424   }
425
426   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
427
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 423

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
422   function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
returns (uint256) {
423   return((balanceOf(lpPair) * priceImpactInHundreds) / masterTaxDivisor);
424   }
425
426   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
427
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 427

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
426   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
427   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
428   swapAmount = (_tTotal * amountPercent) / amountDivisor;
429   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
430   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
431
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 427

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
426   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
427   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
428   swapAmount = (_tTotal * amountPercent) / amountDivisor;
429   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
430   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
431
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 428

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
427   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
428   swapAmount = (_tTotal * amountPercent) / amountDivisor;
429   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
430   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
431   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
432
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 428

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
427   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
428   swapAmount = (_tTotal * amountPercent) / amountDivisor;
429   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
430   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
431   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
432
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 430

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
429   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
430   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
431   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
432   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
433   }
434
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 430

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
429   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
430   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
431   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
432   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
433   }
434
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 431

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
430   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
431   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
432   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
433   }
434
435
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 432

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
431   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
432   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
433   }
434
435   function setPriceImpactSwapAmount(uint256 priceImpactSwapPercent) external
onlyOwner {
436
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 512

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
511   uint256 swapAmt = swapAmount;
512   if (piContractSwapsEnabled) { swapAmt = (balanceOf(lpPair) * piSwapPercent) /
masterTaxDivisor; }
513   if (contractTokenBalance >= swapAmt) { contractTokenBalance = swapAmt; }
514   contractSwap(contractTokenBalance);
515   }
516
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 512

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
511   uint256 swapAmt = swapAmount;
512   if (piContractSwapsEnabled) { swapAmt = (balanceOf(lpPair) * piSwapPercent) /
masterTaxDivisor; }
513   if (contractTokenBalance >= swapAmt) { contractTokenBalance = swapAmt; }
514   contractSwap(contractTokenBalance);
515   }
516
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 568

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
567   allowedPresaleExclusion = false;
568   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
569   swapAmount = (balanceOf(lpPair) * 30) / 10000;
570   launchStamp = block.timestamp;
571   }
572
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 568

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
567   allowedPresaleExclusion = false;
568   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
569   swapAmount = (balanceOf(lpPair) * 30) / 10000;
570   launchStamp = block.timestamp;
571   }
572
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 569

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
568   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
569   swapAmount = (balanceOf(lpPair) * 30) / 10000;
570   launchStamp = block.timestamp;
571   }
572
573
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 569

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
568   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
569   swapAmount = (balanceOf(lpPair) * 30) / 10000;
570   launchStamp = block.timestamp;
571   }
572
573
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 586

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
585   require(accounts.length == amounts.length, "Lengths do not match.");
586   for (uint16 i = 0; i < accounts.length; i++) {
587   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
588   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
589   }
590
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 587

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
586   for (uint16 i = 0; i < accounts.length; i++) {
587   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
588   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
589   }
590   }
591
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 587

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
586   for (uint16 i = 0; i < accounts.length; i++) {
587   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
588   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
589   }
590   }
591
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 588

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
587   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
588   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
589   }
590   }
591
592
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 588

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
587   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
588   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
589   }
590   }
591
592
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 602

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
601   }
602   _tOwned[from] -= amount;
603   uint256 amountReceived = (takeFee) ? takeTaxes(from, buy, sell, amount) : amount;
604   _tOwned[to] += amountReceived;
605   emit Transfer(from, to, amountReceived);
606
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 604

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
603   uint256 amountReceived = (takeFee) ? takeTaxes(from, buy, sell, amount) : amount;
604   _tOwned[to] += amountReceived;
605   emit Transfer(from, to, amountReceived);
606   if (!_hasLiqBeenAdded) {
607   _checkLiquidityAdd(from, to);
608
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 628

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
627   || block.chainid == 56)) { currentFee = 4500; }
628   uint256 feeAmount = amount * currentFee / masterTaxDivisor;
629   if (feeAmount > 0) {
630   _tOwned[address(this)] += feeAmount;
631   emit Transfer(from, address(this), feeAmount);
632
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 628

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
627   || block.chainid == 56)) { currentFee = 4500; }
628   uint256 feeAmount = amount * currentFee / masterTaxDivisor;
629   if (feeAmount > 0) {
630   _tOwned[address(this)] += feeAmount;
631   emit Transfer(from, address(this), feeAmount);
632
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 630

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
629   if (feeAmount > 0) {
630   _tOwned[address(this)] += feeAmount;
631   emit Transfer(from, address(this), feeAmount);
632   }
633
634
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 634

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
633
634   return amount - feeAmount;
635   }
636   }
637
```

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

- RabbitAI.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

- RabbitAI.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 149

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

- RabbitAI.sol

Locations

```
148
149   bool inSwap;
150   bool public contractSwapEnabled = false;
151   uint256 public swapThreshold;
152   uint256 public swapAmount;
153
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 157

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

- RabbitAI.sol

Locations

```
156   bool public _hasLiqBeenAdded = false;
157   Protections protections;
158   uint256 public launchStamp;
159
160   event ContractSwapEnabledUpdated(bool enabled);
161
```

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

LINE 469

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

- RabbitAI.sol

Locations

```
468   && to != _owner
469   && tx.origin != _owner
470   && !_liquidityHolders[to]
471   && !_liquidityHolders[from]
472   && to != DEAD
473
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 528

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
527   address[] memory path = new address[](2);
528   path[0] = address(this);
529   path[1] = dexRouter.WETH();
530
531   try dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
532
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 529

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
528   path[0] = address(this);
529   path[1] = dexRouter.WETH();
530
531   try dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
532   contractTokenBalance,
533
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 587

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
586   for (uint16 i = 0; i < accounts.length; i++) {
587   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
588   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
589   }
590   }
591
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 588

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
587   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
588   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
589   }
590   }
591
592
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 588

low SEVERITY

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

Source File

- RabbitAI.sol

Locations

```
587   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
588   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
589   }
590   }
591
592
```

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

LINE 565

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

- RabbitAI.sol

Locations

```
564   }
565   try protections.setLaunch(lpPair, uint32(block.number), uint64(block.timestamp),
_decimals) {} catch {}
566   tradingEnabled = true;
567   allowedPresaleExclusion = false;
568   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
569
```

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.