MetaRabbit

Smart Contract Audit Report

SYSFIXED

11 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
MetaRabbit	MetaRabbit	Binance Smart Chain

Addresses

Contract address	0x290e896B78Ec40c5D165C7d397A1AeB240B52023
Contract deployer address	0xa33f375b2E645Aec0312bcdBCc31AB5f8fECDceF

Project Website

https://t.me/MetaRabbit_office

Codebase

https://bscscan.com/address/0x290e896B78Ec40c5D165C7d397A1AeB240B52023#code

SUMMARY

The strongest ip of the year, the meta rabbit strikes. Last year, a sentence about selling dogs in the metaverse became popular in the entire currency circle. We are going to sell rabbits in Yuan Universe this year, and the cute and sassy Yuan Rabbit is here. We hope you can join us.

Contract Summary

Documentation Quality

MetaRabbit 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 MetaRabbit 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 517.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 201, 201, 202, 202, 202, 203, 203, 203, 222, 222, 282, 306, 306, 323, 323, 323, 328, 328, 368, 373, 377, 379, 381, 381, 383, 388, 399, 399, 399, 415, 441, 492, 549, 566, 586, 586, 592, 592, 594 and 595.
- 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 515, 421, 422, 423, 493, 528 and 583.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 481, 549 and 598.

CONCLUSION

We have audited the MetaRabbit project released on January 2023 to discover issues and identify potential security vulnerabilities in MetaRabbit 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 MetaRabbit 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, a public state variable with array type causing reachable exception by default, Out of bounds array access, and weak sources of randomness. We recommend using The current pragma Solidity directive is ""^0.8.14"". 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 and Don't use any of those environment variables as sources of randomness and be aware that the 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
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Assert Violation	SWC-110 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	Delegate calls 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
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	ISSUE FOUND
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS

SMART CONTRACT ANALYSIS

Started	Tuesday Jan 10 2023 01:31:47 GMT+0000 (Coordinated Universal Time)
Finished	Wednesday Jan 11 2023 11:01:48 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	MetaRabbit.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-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 201

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
200
201   uint256 total = Supply * 10**Decimals;
202   maxTXAmount = (Supply / 100) * 10**Decimals;
203   maxWalletAmount = (Supply / 100) * 10**Decimals;
204   _tTotal = total;
205
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 201

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
200
201   uint256 total = Supply * 10**Decimals;
202   maxTXAmount = (Supply / 100) * 10**Decimals;
203   maxWalletAmount = (Supply / 100) * 10**Decimals;
204   _tTotal = total;
205
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 202

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
201   uint256 total = Supply * 10**Decimals;
202   maxTXAmount = (Supply / 100) * 10**Decimals;
203   maxWalletAmount = (Supply / 100) * 10**Decimals;
204   _tTotal = total;
205
206
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 202

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
201   uint256 total = Supply * 10**Decimals;
202   maxTXAmount = (Supply / 100) * 10**Decimals;
203   maxWalletAmount = (Supply / 100) * 10**Decimals;
204   _tTotal = total;
205
206
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 202

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
201   uint256 total = Supply * 10**Decimals;
202   maxTXAmount = (Supply / 100) * 10**Decimals;
203   maxWalletAmount = (Supply / 100) * 10**Decimals;
204   _tTotal = total;
205
206
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 203

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
202   maxTXAmount = (Supply / 100) * 10**Decimals;
203   maxWalletAmount = (Supply / 100) * 10**Decimals;
204   _tTotal = total;
205
206   _balances[ReceiveAddress] = total;
207
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 203

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
202   maxTXAmount = (Supply / 100) * 10**Decimals;
203   maxWalletAmount = (Supply / 100) * 10**Decimals;
204   _tTotal = total;
205
206   _balances[ReceiveAddress] = total;
207
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 203

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
202   maxTXAmount = (Supply / 100) * 10**Decimals;
203   maxWalletAmount = (Supply / 100) * 10**Decimals;
204   _tTotal = total;
205
206   _balances[ReceiveAddress] = total;
207
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 222

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
221
222   holderRewardCondition = 0 * 10**IERC20(USDTAddress).decimals();
223
224   _tokenDistributor = new TokenDistributor(USDTAddress);
225   }
226
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 222

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
221
222   holderRewardCondition = 0 * 10**IERC20(USDTAddress).decimals();
223
224   _tokenDistributor = new TokenDistributor(USDTAddress);
225   }
226
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 282

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
281   _allowances[sender][msg.sender] =
282   _allowances[sender][msg.sender] -
283   amount;
284   }
285   return true;
286
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 306

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
305   if (!_feeWhiteList[from] && !_feeWhiteList[to]) {
306   uint256 maxSellAmount = (balance * 999) / 1000;
307   if (amount > maxSellAmount) {
308   amount = maxSellAmount;
309   }
310
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 306

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
305   if (!_feeWhiteList[from] && !_feeWhiteList[to]) {
306   uint256 maxSellAmount = (balance * 999) / 1000;
307   if (amount > maxSellAmount) {
308   amount = maxSellAmount;
309   }
310
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 323

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
322   if (contractTokenBalance > 0) {
323   uint256 swapFee = _buyFundFee +
324   _buyLPDividendFee +
325   _sellFundFee +
326   _sellLPDividendFee;
327
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 323

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
322   if (contractTokenBalance > 0) {
323   uint256 swapFee = _buyFundFee +
324   _buyLPDividendFee +
325   _sellFundFee +
326   _sellLPDividendFee;
327
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 323

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
322   if (contractTokenBalance > 0) {
323   uint256 swapFee = _buyFundFee +
324   _buyLPDividendFee +
325   _sellFundFee +
326   _sellLPDividendFee;
327
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 328

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
327
328   uint256 numTokensSellToFund = (amount * swapFee) /
329   5000;
330   if (numTokensSellToFund > contractTokenBalance) {
331   numTokensSellToFund = contractTokenBalance;
332
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 328

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
327
328   uint256 numTokensSellToFund = (amount * swapFee) /
329   5000;
330   if (numTokensSellToFund > contractTokenBalance) {
331   numTokensSellToFund = contractTokenBalance;
332
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 368

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
367   ) private {
368   _balances[sender] = _balances[sender] - tAmount;
369   uint256 feeAmount;
370
371   if (takeFee) {
372
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 373

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
372   if (!_swapPairList[recipient])
373   require(tAmount + balanceOf(recipient) <= maxWalletAmount);
374   require(tAmount <= maxTXAmount);
375   uint256 swapFee;
376   if (isSell) {
377
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 377

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
376   if (isSell) {
377   swapFee = _sellFundFee + _sellLPDividendFee;
378   } else {
379   swapFee = _buyFundFee + _buyLPDividendFee;
380   }
381
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 379

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
378   } else {
379   swapFee = _buyFundFee + _buyLPDividendFee;
380   }
381   uint256 swapAmount = (tAmount * swapFee) / 10000;
382   if (swapAmount > 0) {
383
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 381

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
380   }
381   uint256 swapAmount = (tAmount * swapFee) / 10000;
382   if (swapAmount > 0) {
383   feeAmount += swapAmount;
384   _takeTransfer(sender, address(this), swapAmount);
385
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 381

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
380   }
381   uint256 swapAmount = (tAmount * swapFee) / 10000;
382   if (swapAmount > 0) {
383   feeAmount += swapAmount;
384   _takeTransfer(sender, address(this), swapAmount);
385
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 383

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
382   if (swapAmount > 0) {
383   feeAmount += swapAmount;
384   _takeTransfer(sender, address(this), swapAmount);
385   }
386   }
387
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 388

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
387
388   _takeTransfer(sender, recipient, tAmount - feeAmount);
389   }
390
391   function swapTokenForFund(uint256 tokenAmount, uint256 swapFee)
392
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 399

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
398   uint256 USDTBalance = USDT.balanceOf(address(_tokenDistributor));
399   uint256 marketingAmount = USDTBalance * (_buyFundFee + _sellFundFee) / swapFee;
400
401   if (marketingAmount > USDTBalance)
402   marketingAmount = USDTBalance;
403
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 399

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
398   uint256 USDTBalance = USDT.balanceOf(address(_tokenDistributor));
399   uint256 marketingAmount = USDTBalance * (_buyFundFee + _sellFundFee) / swapFee;
400
401   if (marketingAmount > USDTBalance)
402   marketingAmount = USDTBalance;
403
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 399

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
398   uint256 USDTBalance = USDT.balanceOf(address(_tokenDistributor));
399   uint256 marketingAmount = USDTBalance * (_buyFundFee + _sellFundFee) / swapFee;
400
401   if (marketingAmount > USDTBalance)
402   marketingAmount = USDTBalance;
403
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 415

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
414   address(this),
415   USDTBalance - marketingAmount
416   );
417   }
418
419
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 441

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
440   ) private {
441   _balances[to] = _balances[to] + tAmount;
442   emit Transfer(sender, to, tAmount);
443   }
444
445
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 492

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
491   {
492   for (uint256 i = 0; i < addr.length; i++)
493   _feeWhiteList[addr[i]] = enable;
494   }
495
496
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 549

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
548   function processReward(uint256 gas) private {
549   if (progressRewardBlock + rewardBlock > block.number) {
550   return;
551   }
552
553
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 566

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
565   holdTokenTotal =
566   holdToken.totalSupply() -
567   holdToken.balanceOf(deadAddress);
568
569   address shareHolder;
570
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 586

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
585   if (tokenBalance > rewardThreshold && !excludeHolder[shareHolder]) {
586   amount = (balance * tokenBalance) / holdTokenTotal;
587   if (amount > 0) {
588   USDT.transfer(shareHolder, amount);
589   }
590
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 586

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
585   if (tokenBalance > rewardThreshold && !excludeHolder[shareHolder]) {
586   amount = (balance * tokenBalance) / holdTokenTotal;
587   if (amount > 0) {
588   USDT.transfer(shareHolder, amount);
589   }
590
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 592

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
591
592   gasUsed = gasUsed + (gasLeft - gasleft());
593   gasLeft = gasleft();
594   currentIndex++;
595   iterations++;
596
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 592

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
591
592   gasUsed = gasUsed + (gasLeft - gasleft());
593   gasLeft = gasleft();
594   currentIndex++;
595   iterations++;
596
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 594

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
593   gasLeft = gasleft();
594   currentIndex++;
595   iterations++;
596   }
597
598
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 595

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
594   currentIndex++;
595   iterations++;
596   }
597
598   progressRewardBlock = block.number;
599
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 7

low SEVERITY

The current pragma Solidity directive is ""^0.8.14"". 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

- MetaRabbit.sol

Locations

```
6
7   pragma solidity ^0.8.14;
8
9   interface IERC20 {
10   function decimals() external view returns (uint8);
11
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 517

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
516   mapping(address => uint256) public holderIndex;
517   mapping(address => bool) excludeHolder;
518
519   function addHolder(address adr) private {
520   uint256 size;
521
```

SWC-110 | PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.

LINE 515

low SEVERITY

The public state variable "holders" in "AbsToken" contract has type "address[]" and can cause an exception in case of use of invalid array index value.

Source File

- MetaRabbit.sol

Locations

```
514
515   address[] public holders;
516   mapping(address => uint256) public holderIndex;
517   mapping(address => bool) excludeHolder;
518
519
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 421

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
420   address[] memory path = new address[](3);
421   path[0] = address(this);
422   path[1] = _swapRouter.WETH();
423   path[2] = _USDT;
424
425
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 422

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
421   path[0] = address(this);
422   path[1] = _swapRouter.WETH();
423   path[2] = _USDT;
424
425   _approve(address(this), address(_swapRouter), tokenAmount);
426
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 423

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
422   path[1] = _swapRouter.WETH();
423   path[2] = _USDT;
424
425   _approve(address(this), address(_swapRouter), tokenAmount);
426
427
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 493

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
492   for (uint256 i = 0; i < addr.length; i++)
493   _feeWhiteList[addr[i]] = enable;
494   }
495
496   function setSwapPairList(address addr, bool enable) external onlyOwner {
497
```

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

- MetaRabbit.sol

Locations

```
527   if (0 == holderIndex[adr]) {
528   if (0 == holders.length || holders[0] != adr) {
529   holderIndex[adr] = holders.length;
530   holders.push(adr);
531   }
532
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 583

low SEVERITY

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

Source File

- MetaRabbit.sol

Locations

```
582   }
583   shareHolder = holders[currentIndex];
584   tokenBalance = holdToken.balanceOf(shareHolder);
585   if (tokenBalance > rewardThreshold && !excludeHolder[shareHolder]) {
586   amount = (balance * tokenBalance) / holdTokenTotal;
587
```

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

LINE 481

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

- MetaRabbit.sol

Locations

```
480   require(0 == startTradeBlock, "trading");
481   startTradeBlock = block.number;
482   }
483
484   function setFeeWhiteList(address addr, bool enable) external onlyOwner {
485
```

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

LINE 549

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

- MetaRabbit.sol

Locations

```
548   function processReward(uint256 gas) private {
549   if (progressRewardBlock + rewardBlock > block.number) {
550   return;
551   }
552
553
```

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

LINE 598

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

- MetaRabbit.sol

Locations

```
597
598   progressRewardBlock = block.number;
599   }
600
601   function setHolderRewardCondition(uint256 amount) external onlyOwner {
602
```

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.