RewardTax

Smart Contract Audit Report

SYSFIXED

25 Dec 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
RewardTax	REWARD	Binance Smart Chain

Addresses

Contract address	0xe552bbA3040D31Aca73880fbd400A70C9B870495
Contract deployer address	0xee48e87f570E4D7D28b6Af21E704713442bC2407

Project Website

https://rewardtax.live/

Codebase

https://bscscan.com/address/0xe552bbA3040D31Aca73880fbd400A70C9B870495#code

SUMMARY

RewardTax is a multi-chain ecosystem including RSWAP, RBRIDGE, Negative tax (Tax as a Reward#TaaR), NFT Marketplace, and Staking on BSC.

Contract Summary

Documentation Quality

RewardTax 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 RewardTax with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 650, 662, 680, 915, 938, 971, 973, 994, 995, 1020, 1022, 1071, 1175, 1207, 1285, 1293, 1294, 1296, 1309, 1314, 1317, 1318, 1321, 1338, 1341, 1342, 1355, 1358, 1359, 1378, 1380, 1394, 1398, 1403, 1403 and 1404.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 10, 82, 329, 392, 419, 504, 589, 707, 737 and 1120.
- SWC-110 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 1163, 1167 and 1313.

CONCLUSION

We have audited the RewardTax project released on December 2022 to discover issues and identify potential security vulnerabilities in RewardTax 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 RewardTax 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 public state variable with array type causing reachable exception by default and out of bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	PASS
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	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.	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

SMART CONTRACT ANALYSIS

Started	Saturday Dec 24 2022 10:39:24 GMT+0000 (Coordinated Universal Time)
Finished	Sunday Dec 25 2022 19:10:54 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	RewardTax.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

SYSFIXED

SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-110	PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.	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-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 650

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
649   ) internal {
650   uint256 newAllowance = token.allowance(address(this), spender) + value;
651   _callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender,
newAllowance));
652   }
653
654
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 662

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
661   require(oldAllowance >= value, "SafeERC20: decreased allowance below zero");
662   uint256 newAllowance = oldAllowance - value;
663   _callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender,
newAllowance));
664   }
665   }
666
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 680

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
679   uint256 nonceAfter = token.nonces(owner);
680   require(nonceAfter == nonceBefore + 1, "SafeERC20: permit did not succeed");
681   }
682
683   /**
684
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 915

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
914   address owner = _msgSender();
915   _approve(owner, spender, allowance(owner, spender) + addedValue);
916   return true;
917   }
918
919
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 938

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
937   unchecked {
938   _approve(owner, spender, currentAllowance - subtractedValue);
939   }
940
941   return true;
942
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 971

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
970   unchecked {
971   _balances[from] = fromBalance - amount;
972   }
973   _balances[to] += amount;
974
975
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 973

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
972   }
973   _balances[to] += amount;
974
975   emit Transfer(from, to, amount);
976
977
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 994

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
993
994   _totalSupply += amount;
995   _balances[account] += amount;
996   emit Transfer(address(0), account, amount);
997
998
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 995

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
994   _totalSupply += amount;
995   _balances[account] += amount;
996   emit Transfer(address(0), account, amount);
997
998   _afterTokenTransfer(address(0), account, amount);
999
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1020

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1019   unchecked {
1020   _balances[account] = accountBalance - amount;
1021   }
1022   _totalSupply -= amount;
1023
1024
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 1022

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1021   }
1022   _totalSupply -= amount;
1023
1024   emit Transfer(account, address(0), amount);
1025
1026
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1071

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1070   unchecked {
1071   _approve(owner, spender, currentAllowance - amount);
1072   }
1073   }
1074   }
1075
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1175

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1174
1175   uint256 constant public DIVISOR = 2**128;
1176
1177   constructor(address taxAccount, uint256 maxTaxAmount, address swapRouterAddress,
address busd, uint256 buyRewardAmount, uint256 preMint) ERC20("RewardTax", "REWARD") {
1178   require(taxAccount != address(0), "taxAccount_ can't be the zero address");
1179
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1207

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1206   emit AddWhitelist(_msgSender());
1207   _mint(msg.sender, preMint * 10 ** decimals());
1208   }
1209
1210   receive() external payable nonReentrant {
1211
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1285

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1284   function _addDividend(uint256 amount, bool _isBuy) internal {
1285   pendingSwapAmount = pendingSwapAmount + amount;
1286
1287   if(pendingSwapAmount > 10000 && !_isBuy && swapEnabled) {
1288   IERC20 rewardToken = IERC20(dividendToken);
1289
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1293

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1292   uint256 updatedBalance = rewardToken.balanceOf(address(this));
1293   uint256 balanceDifference = ((updatedBalance - oldBalance)/2) * DIVISOR;
1294   totalRewardShare = totalRewardShare + (balanceDifference / (totalSupply() -
balanceOf(poolAddress) - balanceOf(address(this)) ));
1295   pendingSwapAmount = 0;
1296   rewardToken.safeTransfer(taxAddress, ((updatedBalance - oldBalance)/2));
1297
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1294

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1293   uint256 balanceDifference = ((updatedBalance - oldBalance)/2) * DIVISOR;
1294   totalRewardShare = totalRewardShare + (balanceDifference / (totalSupply() -
balanceOf(poolAddress) - balanceOf(address(this)) ));
1295   pendingSwapAmount = 0;
1296   rewardToken.safeTransfer(taxAddress, ((updatedBalance - oldBalance)/2));
1297   }
1298
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1296

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1295   pendingSwapAmount = 0;
1296   rewardToken.safeTransfer(taxAddress, ((updatedBalance - oldBalance)/2));
1297   }
1298   catch Error(string memory){
1299
1300
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1309

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1308   if(payoutEnabled) {
1309   for(uint256 i=0; i<userCount; i++) {
1310   if(userCtr == holders.length) {
1311   userCtr = 0;
1312   }
1313
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1314

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1313   address userAddress = holders[userCtr];
1314   uint256 claimableRate = totalRewardShare - rewardShareClaimed[userAddress];
1315   if(claimableRate > 0) {
1316   uint256 userBalance = IERC20(address(this)).balanceOf(userAddress);
1317   rewardShareClaimed[userAddress] = rewardShareClaimed[userAddress] + claimableRate;
1318
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1317

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1316   uint256 userBalance = IERC20(address(this)).balanceOf(userAddress);
1317   rewardShareClaimed[userAddress] = rewardShareClaimed[userAddress] + claimableRate;
1318   IERC20(dividendToken).safeTransfer(userAddress ,(userBalance *
claimableRate)/DIVISOR);
1319
1320   }
1321
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1318

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1317   rewardShareClaimed[userAddress] = rewardShareClaimed[userAddress] + claimableRate;
1318   IERC20(dividendToken).safeTransfer(userAddress ,(userBalance *
claimableRate)/DIVISOR);
1319
1320   }
1321   userCtr++;
1322
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1321

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1320   }
1321   userCtr++;
1322   }
1323   }
1324
1325
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1338

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1337   //do the claim for the user
1338   uint256 claimableRate = totalRewardShare - rewardShareClaimed[userAddress];
1339   uint256 userBalance = IERC20(address(this)).balanceOf(userAddress);
1340   if(claimableRate > 0 && userBalance > 0) {
1341   rewardShareClaimed[userAddress] = rewardShareClaimed[userAddress] + claimableRate;
1342
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1341

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1340   if(claimableRate > 0 && userBalance > 0) {
1341   rewardShareClaimed[userAddress] = rewardShareClaimed[userAddress] + claimableRate;
1342   IERC20(dividendToken).safeTransfer(userAddress ,(userBalance *
claimableRate)/DIVISOR);
1343   }
1344   }
1345
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1342

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1341   rewardShareClaimed[userAddress] = rewardShareClaimed[userAddress] + claimableRate;
1342   IERC20(dividendToken).safeTransfer(userAddress ,(userBalance *
claimableRate)/DIVISOR);
1343   }
1344   }
1345   else{
1346
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1355

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1354   //do the claim for the user
1355   uint256 claimableRate = totalRewardShare - rewardShareClaimed[userAddress];
1356   uint256 userBalance = IERC20(address(this)).balanceOf(userAddress);
1357   if(claimableRate > 0 && userBalance > 0) {
1358   rewardShareClaimed[userAddress] = rewardShareClaimed[userAddress] + claimableRate;
1359
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1358

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1357   if(claimableRate > 0 && userBalance > 0) {
1358   rewardShareClaimed[userAddress] = rewardShareClaimed[userAddress] + claimableRate;
1359   IERC20(dividendToken).safeTransfer(userAddress ,(userBalance *
claimableRate)/DIVISOR);
1360   }
1361   }
1362
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1359

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1358   rewardShareClaimed[userAddress] = rewardShareClaimed[userAddress] + claimableRate;
1359   IERC20(dividendToken).safeTransfer(userAddress ,(userBalance *
claimableRate)/DIVISOR);
1360   }
1361   }
1362
1363
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1378

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1377   function _rewardUser(address userAddress, uint256 amount) internal {
1378   uint256 rewardAmount = (buyReward * amount)/100;
1379   //send user the reward from current address
1380   if(balanceOf(address(this)) - pendingSwapAmount >= rewardAmount) {
1381   _transfer(address(this), userAddress, rewardAmount);
1382
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1380

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1379   //send user the reward from current address
1380   if(balanceOf(address(this)) - pendingSwapAmount >= rewardAmount) {
1381   _transfer(address(this), userAddress, rewardAmount);
1382   emit RewardUser(userAddress, rewardAmount);
1383   }
1384
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1394

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1393   require(amount >= 10000, "Amount too low");
1394   uint256 taxAmount = (buyTax * amount) / 100;
1395   _transfer(from , address(this), taxAmount);
1396   _addDividend(taxAmount, true);
1397   _rewardUser(to, amount);
1398
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1398

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1397   _rewardUser(to, amount);
1398   amount = amount - taxAmount;
1399   }
1400   else if(to == poolAddress) {
1401   //Add sellTax
1402
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1403

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1402   require(amount >= 10000, "Amount too low");
1403   uint256 taxAmount = (sellTax * amount) / 100;
1404   amount = amount - taxAmount;
1405   _transfer(from , address(this), taxAmount);
1406   _addDividend(taxAmount, false);
1407
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1404

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1403   uint256 taxAmount = (sellTax * amount) / 100;
1404   amount = amount - taxAmount;
1405   _transfer(from , address(this), taxAmount);
1406   _addDividend(taxAmount, false);
1407   }
1408
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 10

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- RewardTax.sol

Locations

```
9
10   pragma solidity ^0.8.0;
11
12   /**
13   * @dev Contract module that helps prevent reentrant calls to a function.
14
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 82

low SEVERITY

The current pragma Solidity directive is ""^0.8.1"". 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

- RewardTax.sol

Locations

```
81
82   pragma solidity ^0.8.1;
83
84   /**
85   * @dev Collection of functions related to the address type
86
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 329

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- RewardTax.sol

Locations

```
328
329   pragma solidity ^0.8.0;
330
331   /**
332    * @dev Interface of the ERC20 Permit extension allowing approvals to be made via
signatures, as defined in
333
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 392

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- RewardTax.sol

Locations

```
391
392   pragma solidity ^0.8.0;
393
394   /**
395   * @dev Provides information about the current execution context, including the
396
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 419

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- RewardTax.sol

Locations

```
418
419   pragma solidity ^0.8.0;
420
421
422   /**
423
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 504

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- RewardTax.sol

Locations

```
503
504   pragma solidity ^0.8.0;
505
506   /**
507   * @dev Interface of the ERC20 standard as defined in the EIP.
508
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 589

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- RewardTax.sol

Locations

```
588
589   pragma solidity ^0.8.0;
590
591
592
593
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 707

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- RewardTax.sol

Locations

```
706
707   pragma solidity ^0.8.0;
708
709
710   /**
711
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 737

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- RewardTax.sol

Locations

```
736
737   pragma solidity ^0.8.0;
738
739
740
741
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1120

low SEVERITY

The current pragma Solidity directive is ""^0.8.4"". 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

- RewardTax.sol

Locations

```
1119
1120   pragma solidity ^0.8.4;
1121
1122
1123
1124
```

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

LINE 1163

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1162   bool public payoutEnabled;
1163   address[] public swapPath;
1164
1165   mapping(address => uint256) public rewardShareClaimed;
1166
1167
```

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

LINE 1167

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1166
1167   address[] public holders;
1168   mapping(address => bool) public isHolder;
1169   address public poolAddress;
1170   mapping(address => bool) public blacklist;
1171
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1313

low SEVERITY

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

Source File

- RewardTax.sol

Locations

```
1312   }
1313   address userAddress = holders[userCtr];
1314   uint256 claimableRate = totalRewardShare - rewardShareClaimed[userAddress];
1315   if(claimableRate > 0) {
1316   uint256 userBalance = IERC20(address(this)).balanceOf(userAddress);
1317
```

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.