SugarYield

Smart Contract Audit Report

SYSFIXED

25 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
SugarYield	SUGAR	Binance Smart Chain

Addresses

Contract address	0x57528b45134f09F2e0069334a36A7e14AF74745F
Contract deployer address	0xcadbe33ec806a88059FEa4a0F098EAd3afe05c4E

Project Website

https://sugaryield.com/

Codebase

https://bscscan.com/address/0x57528b45134f09F2e0069334a36A7e14AF74745F#code

SUMMARY

SugarYield.com is a DeFi insurance protocol that allows third-party participants to speculate on the performance of underlying pegged assets depending on their performance histories, including BUSD, DAI,USDT and many other stable tokens.

Contract Summary

Documentation Quality

SugarYield 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 SugarYield 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 128, 138, 146, 165, 167, 179, 180, 194, 196, 506, 507, 507, 552, 552, 552, 553, 553, 604, 611, 611, 658, 685, 685, 689, 690, 690, 690, 714, 714, 716, 727, 728, 743, 771, 773, 776, 776, 781, 781, 789, 815, 815, 853, 853, 857 and 857.
- 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 733, 734, 761 and 762.

CONCLUSION

We have audited the SugarYield project released on January 2023 to discover issues and identify potential security vulnerabilities in SugarYield 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 SugarYield 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, 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.	PASS
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	PASS
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

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

SMART CONTRACT ANALYSIS

Started	Tuesday Jan 24 2023 06:54:38 GMT+0000 (Coordinated Universal Time)
Finished	Wednesday Jan 25 2023 10:04:05 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	SugarYield.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 128

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
127   unchecked {
128   _approve(sender, _msgSender(), currentAllowance - amount);
129   }
130   }
131
132
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 138

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
137   function increaseAllowance(address spender, uint256 addedValue) public virtual
returns (bool) {
138   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
139   return true;
140   }
141
142
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 146

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
145   unchecked {
146   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
147   }
148
149   return true;
150
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 165

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
164   unchecked {
165   _balances[sender] = senderBalance - amount;
166   }
167   _balances[recipient] += amount;
168
169
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 167

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
166   }
167   _balances[recipient] += amount;
168
169   emit Transfer(sender, recipient, amount);
170
171
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 179

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
178
179   _totalSupply += amount;
180   _balances[account] += amount;
181   emit Transfer(address(0), account, amount);
182
183
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 180

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
179   _totalSupply += amount;
180   _balances[account] += amount;
181   emit Transfer(address(0), account, amount);
182
183   _afterTokenTransfer(address(0), account, amount);
184
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 194

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
193   unchecked {
194   _balances[account] = accountBalance - amount;
195   }
196   _totalSupply -= amount;
197
198
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 196

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
195   }
196   _totalSupply -= amount;
197
198   emit Transfer(account, address(0), amount);
199
200
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 506

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
505   {
506   require(_arg.buyFee_ + _arg.sellFee_ <= 25, "Total buy and sell fees cannot be more
than 25%");
507   require(_arg.marketingShare_ + _arg.liquidityShare_ + _arg.charityShare_ == 100,
"Total fee shares must be equal to 100");
508   require(_arg.maxTransactionRateBuy_ >= 1 && _arg.maxTransactionRateSell_ >= 1, "Max
transfer rates must be greater than 0.1%");
509   require(_arg.maxWalletLimitRate_ >= 10, "Max wallet limit rate must be greater than
1%");
510
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 507

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
506   require(_arg.buyFee_ + _arg.sellFee_ <= 25, "Total buy and sell fees cannot be more
than 25%");
507   require(_arg.marketingShare_ + _arg.liquidityShare_ + _arg.charityShare_ == 100,
"Total fee shares must be equal to 100");
508   require(_arg.maxTransactionRateBuy_ >= 1 && _arg.maxTransactionRateSell_ >= 1, "Max
transfer rates must be greater than 0.1%");
509   require(_arg.maxWalletLimitRate_ >= 10, "Max wallet limit rate must be greater than
1%");
510
511
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 507

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
506   require(_arg.buyFee_ + _arg.sellFee_ <= 25, "Total buy and sell fees cannot be more
than 25%");
507   require(_arg.marketingShare_ + _arg.liquidityShare_ + _arg.charityShare_ == 100,
"Total fee shares must be equal to 100");
508   require(_arg.maxTransactionRateBuy_ >= 1 && _arg.maxTransactionRateSell_ >= 1, "Max
transfer rates must be greater than 0.1%");
509   require(_arg.maxWalletLimitRate_ >= 10, "Max wallet limit rate must be greater than
1%");
510
511
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 552

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
551
552   swapTokensAtAmount = _arg.totalSupply_ * (10 ** 18) / 5000;
553   _mint(owner(), _arg.totalSupply_ * (10 ** 18));
554
555   emit TokenCreated(owner(), address(this), TokenClass.basicToken, 3);
556
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 552

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
551
552   swapTokensAtAmount = _arg.totalSupply_ * (10 ** 18) / 5000;
553   _mint(owner(), _arg.totalSupply_ * (10 ** 18));
554
555   emit TokenCreated(owner(), address(this), TokenClass.basicToken, 3);
556
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 552

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
551
552   swapTokensAtAmount = _arg.totalSupply_ * (10 ** 18) / 5000;
553   _mint(owner(), _arg.totalSupply_ * (10 ** 18));
554
555   emit TokenCreated(owner(), address(this), TokenClass.basicToken, 3);
556
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 553

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
552   swapTokensAtAmount = _arg.totalSupply_ * (10 ** 18) / 5000;
553   _mint(owner(), _arg.totalSupply_ * (10 ** 18));
554
555   emit TokenCreated(owner(), address(this), TokenClass.basicToken, 3);
556   }
557
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 553

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
552   swapTokensAtAmount = _arg.totalSupply_ * (10 ** 18) / 5000;
553   _mint(owner(), _arg.totalSupply_ * (10 ** 18));
554
555   emit TokenCreated(owner(), address(this), TokenClass.basicToken, 3);
556   }
557
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 604

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
603   function updateFees(uint256 _buyFee, uint256 _sellFee) external onlyOwner {
604   require(_buyFee + _sellFee <= 25, "Total buy and sell fees cannot be more than
25%");
605   buyFee = _buyFee;
606   sellFee = _sellFee;
607   emit FeesUpdated(buyFee, sellFee);
608
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 611

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
610   function updateFeeShares(uint256 _marketingFeeShare, uint256 _liquidityFeeShare,
uint256 _charityShare) external onlyOwner {
611   require(_marketingFeeShare + _liquidityFeeShare + _charityShare == 100, "Total fee
shares must be equal to 100");
612   marketingShare = _marketingFeeShare;
613   liquidityShare = _liquidityFeeShare;
614   charityShare   = _charityShare;
615
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 611

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
610   function updateFeeShares(uint256 _marketingFeeShare, uint256 _liquidityFeeShare,
uint256 _charityShare) external onlyOwner {
611   require(_marketingFeeShare + _liquidityFeeShare + _charityShare == 100, "Total fee
shares must be equal to 100");
612   marketingShare = _marketingFeeShare;
613   liquidityShare = _liquidityFeeShare;
614   charityShare   = _charityShare;
615
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 658

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
657   uint balance  = balanceOf(to);
658   require(balance + amount <= maxWalletAmount(), "MaxWallet: Transfer amount exceeds
the maxWalletAmount");
659   }
660   }
661
662
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 685

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
684   if(liquidityShare > 0) {
685   uint256 liquidityTokens = contractTokenBalance * liquidityShare / 100;
686   swapAndLiquify(liquidityTokens);
687   }
688
689
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 685

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
684   if(liquidityShare > 0) {
685   uint256 liquidityTokens = contractTokenBalance * liquidityShare / 100;
686   swapAndLiquify(liquidityTokens);
687   }
688
689
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 689

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
688
689   if(marketingShare + charityShare > 0) {
690   uint256 feeTokens = (contractTokenBalance * (marketingShare + charityShare)) / 100;
691   swapAndSendFees(feeTokens);
692   }
693
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 690

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
689   if(marketingShare + charityShare > 0) {
690   uint256 feeTokens = (contractTokenBalance * (marketingShare + charityShare)) / 100;
691   swapAndSendFees(feeTokens);
692   }
693
694
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 690

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
689   if(marketingShare + charityShare > 0) {
690   uint256 feeTokens = (contractTokenBalance * (marketingShare + charityShare)) / 100;
691   swapAndSendFees(feeTokens);
692   }
693
694
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 690

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
689   if(marketingShare + charityShare > 0) {
690   uint256 feeTokens = (contractTokenBalance * (marketingShare + charityShare)) / 100;
691   swapAndSendFees(feeTokens);
692   }
693
694
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 714

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
713   }
714   uint256 fees = amount * _totalFees / 100;
715
716   amount = amount - fees;
717
718
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 714

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
713   }
714   uint256 fees = amount * _totalFees / 100;
715
716   amount = amount - fees;
717
718
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 716

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
715
716   amount = amount - fees;
717
718   super._transfer(from, address(this), fees);
719   }
720
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 727

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
726   function swapAndLiquify(uint256 tokens) private {
727   uint256 half = tokens / 2;
728   uint256 otherHalf = tokens - half;
729
730   uint256 initialBalance = address(this).balance;
731
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 728

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
727   uint256 half = tokens / 2;
728   uint256 otherHalf = tokens - half;
729
730   uint256 initialBalance = address(this).balance;
731
732
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 743

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
742
743   uint256 newBalance = address(this).balance - initialBalance;
744
745   uniswapV2Router.addLiquidityETH{value: newBalance}(
746   address(this),
747
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 771

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
770
771   uint256 newBalance = address(this).balance - initialBalance;
772
773   uint256 bnbShare = marketingShare + charityShare;
774
775
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 773

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
772
773   uint256 bnbShare = marketingShare + charityShare;
774
775   if(marketingShare > 0) {
776   uint256 marketingBnb = newBalance * marketingShare / bnbShare;
777
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 776

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
775   if(marketingShare > 0) {
776   uint256 marketingBnb = newBalance * marketingShare / bnbShare;
777   sendBNB(payable(marketingWallet), marketingBnb);
778   }
779
780
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 776

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
775   if(marketingShare > 0) {
776   uint256 marketingBnb = newBalance * marketingShare / bnbShare;
777   sendBNB(payable(marketingWallet), marketingBnb);
778   }
779
780
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 781

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
780   if(charityShare > 0) {
781   uint256 charityBnb = newBalance * charityShare / bnbShare;
782   sendBNB(payable(charityWallet), charityBnb);
783   }
784
785
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 781

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
780   if(charityShare > 0) {
781   uint256 charityBnb = newBalance * charityShare / bnbShare;
782   sendBNB(payable(charityWallet), charityBnb);
783   }
784
785
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 789

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
788   function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner{
789   require(newAmount > totalSupply() / 100000, "SwapTokensAtAmount must be greater
than 0.001% of total supply");
790   swapTokensAtAmount = newAmount;
791   }
792
793
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 815

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
814   function maxWalletAmount() public view returns (uint256) {
815   return totalSupply() * maxWalletLimitRate / 1000;
816   }
817
818   function setMaxWalletRate_Denominator1000(uint256 _val) external onlyOwner
_maxWalletAvailable {
819
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 815

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
814   function maxWalletAmount() public view returns (uint256) {
815   return totalSupply() * maxWalletLimitRate / 1000;
816   }
817
818   function setMaxWalletRate_Denominator1000(uint256 _val) external onlyOwner
_maxWalletAvailable {
819
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 853

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
852   function maxTransferAmountBuy() public view returns (uint256) {
853   return totalSupply() * maxTransactionRateBuy / 1000;
854   }
855
856   function maxTransferAmountSell() public view returns (uint256) {
857
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 853

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
852   function maxTransferAmountBuy() public view returns (uint256) {
853   return totalSupply() * maxTransactionRateBuy / 1000;
854   }
855
856   function maxTransferAmountSell() public view returns (uint256) {
857
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 857

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
856   function maxTransferAmountSell() public view returns (uint256) {
857   return totalSupply() * maxTransactionRateSell / 1000;
858   }
859
860   function setMaxTransactionRates_Denominator1000(uint256 _maxTransactionRateBuy,
uint256 _maxTransactionRateSell) external onlyOwner _maxTransactionAvailable {
861
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 857

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
856   function maxTransferAmountSell() public view returns (uint256) {
857   return totalSupply() * maxTransactionRateSell / 1000;
858   }
859
860   function setMaxTransactionRates_Denominator1000(uint256 _maxTransactionRateBuy,
uint256 _maxTransactionRateSell) external onlyOwner _maxTransactionAvailable {
861
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 733

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
732   address[] memory path = new address[](2);
733   path[0] = address(this);
734   path[1] = uniswapV2Router.WETH();
735
736   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
737
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 734

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
733   path[0] = address(this);
734   path[1] = uniswapV2Router.WETH();
735
736   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
737   half,
738
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 761

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
760   address[] memory path = new address[](2);
761   path[0] = address(this);
762   path[1] = uniswapV2Router.WETH();
763
764   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
765
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 762

low SEVERITY

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

Source File

- SugarYield.sol

Locations

```
761   path[0] = address(this);
762   path[1] = uniswapV2Router.WETH();
763
764   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
765   tokenAmount,
766
```

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.