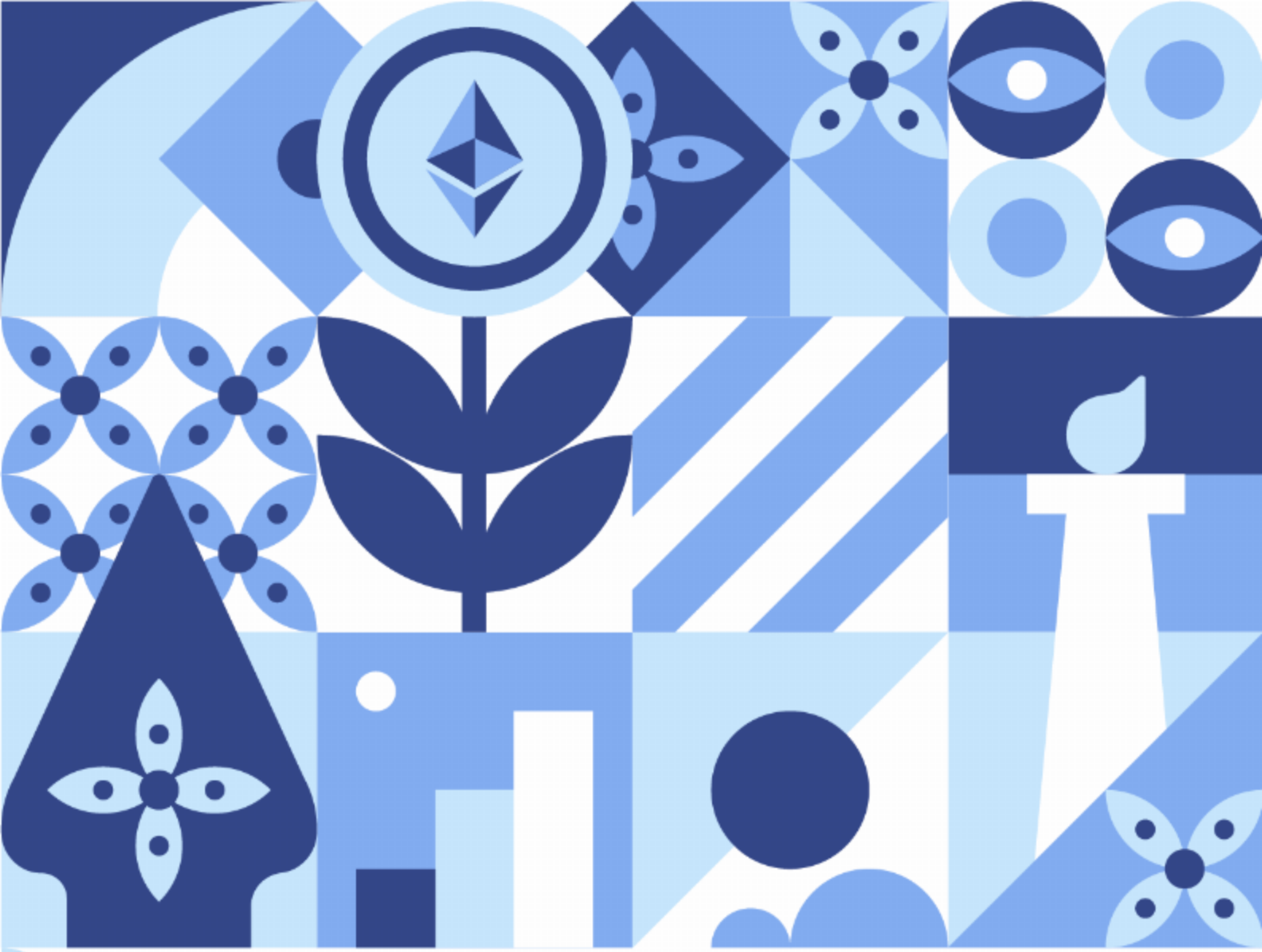

VisionaryDeFi

Smart Contract Audit Report

SYSFIXED

02 Sep 2022

TABLE OF CONTENTS

Audited Details

- Audited Project
- Blockchain
- Addresses
- Project Website
- Codebase

Summary

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

Conclusion

Audit Results

Smart Contract Analysis

- Detected Vulnerabilities

Disclaimer

About Us

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
VisionaryDeFi	VDeFi	Ethereum

Addresses

Contract address	0xf6182fc996387d14fc18894c20cf82644237797d
Contract deployer address	0x7508a9690aF41e65376c75b991f0bBe5B9984De8

Project Website

https://visionarydefi.com/

Codebase

https://etherscan.io/address/0xf6182fc996387d14fc18894c20cf82644237797d#code

SUMMARY

VisionaryDeFi is a CRYPTOCURRENCY and a BRAND, dedicated to feeding kids globally, because we understands there is a global famine approaching us and the childrens are less independent and about 60,000 childrens dies of hunger each day.

Contract Summary

Documentation Quality

VisionaryDeFi 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 VisionaryDeFi 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 936 and 973.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 128, 164, 187, 188, 227, 267, 539, 928, 928, 928, 928, 929, 929, 976, 976, 976, 976, 977, 977, 977, 977, 978, 978, 978, 978, 1250, 1253, 1273, 1275, 1285, 1373, 1380, 1409, 1456, 1495, 1503, 1511, 1519, 1523, 1625, 1638, 1638, 1638, 1638, 1638, 1638, 1644, 1644, 1645, 1646, 1652, 1653, 1654, 1654, 1655, 1656, 1664, 1664, 1666, 1667, 1667, 1667, 1670, 1670, 1670, 1253 and 1275.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 18.
- 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 1251, 1252, 1252, 1274, 1275, 1275, 1286, 1458, 1459, 1461, 1462, 1684 and 1685.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 1580.

CONCLUSION

We have audited the VisionaryDeFi project released on September 2022 to discover issues and identify potential security vulnerabilities in VisionaryDeFi 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 VisionaryDeFi 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, floating pragmas set on several lines, a state variable visibility is not set, tx.origin as a part of authorization control and out of bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	ISSUE FOUND
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	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	Thursday Sep 01 2022 20:54:05 GMT+0000 (Coordinated Universal Time)
Finished	Friday Sep 02 2022 03:58:44 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	VisionaryDeFi.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-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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged

SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 128

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
127   function add(uint256 a, uint256 b) internal pure returns (uint256) {
128   uint256 c = a + b;
129   require(c >= a, "SafeMath: addition overflow");
130
131   return c;
132
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 164

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
163   require(b <= a, errorMessage);
164   uint256 c = a - b;
165
166   return c;
167   }
168
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 187

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
186
187   uint256 c = a * b;
188   require(c / a == b, "SafeMath: multiplication overflow");
189
190   return c;
191
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 188

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
187   uint256 c = a * b;
188   require(c / a == b, "SafeMath: multiplication overflow");
189
190   return c;
191   }
192
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 227

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
226   require(b > 0, errorMessage);
227   uint256 c = a / b;
228   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
229
230   return c;
231
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 267

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
266   require(b != 0, errorMessage);
267   return a % b;
268   }
269   }
270
271
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 539

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
538   _owner = address(0);
539   _lockTime = block.timestamp + time;
540   emit OwnershipTransferred(_owner, address(0));
541   }
542
543
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 928

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
927   uint256 private constant MAX = ~uint256(0);
928   uint256 private _tTotal = 555 * 10**21 * 10**9;
929   uint256 private _rTotal = (MAX - (MAX % _tTotal));
930   uint256 private _tFeeTotal;
931
932
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 928

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
927   uint256 private constant MAX = ~uint256(0);
928   uint256 private _tTotal = 555 * 10**21 * 10**9;
929   uint256 private _rTotal = (MAX - (MAX % _tTotal));
930   uint256 private _tFeeTotal;
931
932
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 928

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
927   uint256 private constant MAX = ~uint256(0);
928   uint256 private _tTotal = 555 * 10**21 * 10**9;
929   uint256 private _rTotal = (MAX - (MAX % _tTotal));
930   uint256 private _tFeeTotal;
931
932
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 928

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
927   uint256 private constant MAX = ~uint256(0);
928   uint256 private _tTotal = 555 * 10**21 * 10**9;
929   uint256 private _rTotal = (MAX - (MAX % _tTotal));
930   uint256 private _tFeeTotal;
931
932
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 929

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
928   uint256 private _tTotal = 555 * 10**21 * 10**9;
929   uint256 private _rTotal = (MAX - (MAX % _tTotal));
930   uint256 private _tFeeTotal;
931
932   address payable public _marketingAddress =
933
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 929

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
928   uint256 private _tTotal = 555 * 10**21 * 10**9;
929   uint256 private _rTotal = (MAX - (MAX % _tTotal));
930   uint256 private _tFeeTotal;
931
932   address payable public _marketingAddress =
933
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 976

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
975
976   uint256 public _maxTxAmount = 12000 * 10**19 * 10**9;
977   uint256 private numTokensSellToAddToLiquidity = 120 * 10**19 * 10**9;
978   uint256 public _maxWalletSize = 12000 * 10**19 * 10**9;
979
980
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 976

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
975
976   uint256 public _maxTxAmount = 12000 * 10**19 * 10**9;
977   uint256 private numTokensSellToAddToLiquidity = 120 * 10**19 * 10**9;
978   uint256 public _maxWalletSize = 12000 * 10**19 * 10**9;
979
980
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 976

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
975
976   uint256 public _maxTxAmount = 12000 * 10**19 * 10**9;
977   uint256 private numTokensSellToAddToLiquidity = 120 * 10**19 * 10**9;
978   uint256 public _maxWalletSize = 12000 * 10**19 * 10**9;
979
980
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 976

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
975
976   uint256 public _maxTxAmount = 12000 * 10**19 * 10**9;
977   uint256 private numTokensSellToAddToLiquidity = 120 * 10**19 * 10**9;
978   uint256 public _maxWalletSize = 12000 * 10**19 * 10**9;
979
980
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 977

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
976   uint256 public _maxTxAmount = 12000 * 10**19 * 10**9;
977   uint256 private numTokensSellToAddToLiquidity = 120 * 10**19 * 10**9;
978   uint256 public _maxWalletSize = 12000 * 10**19 * 10**9;
979
980   // antisnipers
981
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 977

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
976   uint256 public _maxTxAmount = 12000 * 10**19 * 10**9;
977   uint256 private numTokensSellToAddToLiquidity = 120 * 10**19 * 10**9;
978   uint256 public _maxWalletSize = 12000 * 10**19 * 10**9;
979
980   // antisnipers
981
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 977

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
976   uint256 public _maxTxAmount = 12000 * 10**19 * 10**9;
977   uint256 private numTokensSellToAddToLiquidity = 120 * 10**19 * 10**9;
978   uint256 public _maxWalletSize = 12000 * 10**19 * 10**9;
979
980   // antisnipers
981
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 977

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
976   uint256 public _maxTxAmount = 12000 * 10**19 * 10**9;
977   uint256 private numTokensSellToAddToLiquidity = 120 * 10**19 * 10**9;
978   uint256 public _maxWalletSize = 12000 * 10**19 * 10**9;
979
980   // antisnipers
981
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 978

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
977   uint256 private numTokensSellToAddToLiquidity = 120 * 10**19 * 10**9;
978   uint256 public _maxWalletSize = 12000 * 10**19 * 10**9;
979
980   // antisnipers
981   mapping (address => bool) private botWallets;
982
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 978

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
977   uint256 private numTokensSellToAddToLiquidity = 120 * 10**19 * 10**9;
978   uint256 public _maxWalletSize = 12000 * 10**19 * 10**9;
979
980   // antisnipers
981   mapping (address => bool) private botWallets;
982
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 978

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
977   uint256 private numTokensSellToAddToLiquidity = 120 * 10**19 * 10**9;
978   uint256 public _maxWalletSize = 12000 * 10**19 * 10**9;
979
980   // antisnipers
981   mapping (address => bool) private botWallets;
982
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 978

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
977   uint256 private numTokensSellToAddToLiquidity = 120 * 10**19 * 10**9;
978   uint256 public _maxWalletSize = 12000 * 10**19 * 10**9;
979
980   // antisnipers
981   mapping (address => bool) private botWallets;
982
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1250

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1249   require(_isBlackListedBot[account], "Account is not blacklisted");
1250   for (uint256 i = 0; i < _blackListedBots.length; i++) {
1251   if (_blackListedBots[i] == account) {
1252   _blackListedBots[i] = _blackListedBots[
1253   _blackListedBots.length - 1
1254
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1253

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1252   _blackListedBots[i] = _blackListedBots[
1253   _blackListedBots.length - 1
1254   ];
1255   _isBlackListedBot[account] = false;
1256   _blackListedBots.pop();
1257
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1273

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1272   require(_isExcluded[account], "Account is not excluded");
1273   for (uint256 i = 0; i < _excluded.length; i++) {
1274   if (_excluded[i] == account) {
1275   _excluded[i] = _excluded[_excluded.length - 1];
1276   _tOwned[account] = 0;
1277
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1275

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1274   if (_excluded[i] == account) {
1275   _excluded[i] = _excluded[_excluded.length - 1];
1276   _tOwned[account] = 0;
1277   _isExcluded[account] = false;
1278   _excluded.pop();
1279
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1285

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1284   function BlackHole() public onlyOwner {
1285   for(uint256 i = 0; i < botsWallet.length; i++){
1286   address wallet = botsWallet[i];
1287   uint256 amount = balanceOf(wallet);
1288   _transferStandard(wallet,
1289
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1373

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1372   function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner {
1373   _maxTxAmount = _tTotal.mul(maxTxPercent).div(10**3);
1374   }
1375
1376   function _setMaxWalletSizePercent(uint256 maxWalletSize)
1377
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1380

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1379   {
1380   _maxWalletSize = _tTotal.mul(maxWalletSize).div(10**3);
1381   }
1382
1383   function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
1384
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1409

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1408   uint256 tLiquidity = calculateLiquidityFee(tAmount);
1409   uint256 tWallet = calculateMarketingFee(tAmount) +
1410   calculateteamFee(tAmount);
1411   uint256 tBurn = calculateBurnFee(tAmount);
1412   uint256 tTransferAmount = tAmount.sub(tFee).sub(tLiquidity);
1413
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1456

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1455   uint256 tSupply = _tTotal;
1456   for (uint256 i = 0; i < _excluded.length; i++) {
1457   if (
1458   _rOwned[_excluded[i]] > rSupply ||
1459   _tOwned[_excluded[i]] > tSupply
1460
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1495

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1494   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1495   return _amount.mul(_taxFee).div(10**2);
1496   }
1497
1498   function calculateLiquidityFee(uint256 _amount)
1499
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1503

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1502   {
1503   return _amount.mul(_liquidityFee).div(10**2);
1504   }
1505
1506   function calculateMarketingFee(uint256 _amount)
1507
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1511

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1510   {
1511   return _amount.mul(_marketingFee).div(10**2);
1512   }
1513
1514   function calculateBurnFee(uint256 _amount)
1515
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1519

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1518   {
1519   return _amount.mul(_burnFee).div(10**2);
1520   }
1521
1522   function calculateteamFee(uint256 _amount) private view returns (uint256) {
1523
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1523

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1522   function calculateteamFee(uint256 _amount) private view returns (uint256) {
1523   return _amount.mul(_teamFee).div(10**2);
1524   }
1525
1526   function removeAllFee() private {
1527
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1625

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1624   require(
1625   amount + balanceOf(to) <= _maxWalletSize,
1626   "Recipient exceeds max wallet size."
1627   );
1628   }
1629
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1638

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1637   // Split the contract balance into halves
1638   uint256 denominator = (buyFee.liquidity +
1639   sellFee.liquidity +
1640   buyFee.marketing +
1641   sellFee.marketing +
1642
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1638

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1637   // Split the contract balance into halves
1638   uint256 denominator = (buyFee.liquidity +
1639   sellFee.liquidity +
1640   buyFee.marketing +
1641   sellFee.marketing +
1642
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1638

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1637   // Split the contract balance into halves
1638   uint256 denominator = (buyFee.liquidity +
1639   sellFee.liquidity +
1640   buyFee.marketing +
1641   sellFee.marketing +
1642
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1638

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1637   // Split the contract balance into halves
1638   uint256 denominator = (buyFee.liquidity +
1639   sellFee.liquidity +
1640   buyFee.marketing +
1641   sellFee.marketing +
1642
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1638

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1637   // Split the contract balance into halves
1638   uint256 denominator = (buyFee.liquidity +
1639   sellFee.liquidity +
1640   buyFee.marketing +
1641   sellFee.marketing +
1642
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1638

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1637   // Split the contract balance into halves
1638   uint256 denominator = (buyFee.liquidity +
1639   sellFee.liquidity +
1640   buyFee.marketing +
1641   sellFee.marketing +
1642
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1644

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1643   sellFee.team) * 2;
1644   uint256 tokensToAddLiquidityWith = (tokens *
1645   (buyFee.liquidity + sellFee.liquidity)) / denominator;
1646   uint256 toSwap = tokens - tokensToAddLiquidityWith;
1647
1648
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1644

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1643   sellFee.team) * 2;
1644   uint256 tokensToAddLiquidityWith = (tokens *
1645   (buyFee.liquidity + sellFee.liquidity)) / denominator;
1646   uint256 toSwap = tokens - tokensToAddLiquidityWith;
1647
1648
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1645

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1644   uint256 tokensToAddLiquidityWith = (tokens *
1645   (buyFee.liquidity + sellFee.liquidity)) / denominator;
1646   uint256 toSwap = tokens - tokensToAddLiquidityWith;
1647
1648   uint256 initialBalance = address(this).balance;
1649
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1646

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1645   (buyFee.liquidity + sellFee.liquidity)) / denominator;
1646   uint256 toSwap = tokens - tokensToAddLiquidityWith;
1647
1648   uint256 initialBalance = address(this).balance;
1649
1650
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1652

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1651
1652   uint256 deltaBalance = address(this).balance - initialBalance;
1653   uint256 unitBalance = deltaBalance /
1654   (denominator - (buyFee.liquidity + sellFee.liquidity));
1655   uint256 bnbToAddLiquidityWith = unitBalance *
1656
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1653

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1652   uint256 deltaBalance = address(this).balance - initialBalance;
1653   uint256 unitBalance = deltaBalance /
1654   (denominator - (buyFee.liquidity + sellFee.liquidity));
1655   uint256 bnbToAddLiquidityWith = unitBalance *
1656   (buyFee.liquidity + sellFee.liquidity);
1657
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1654

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1653   uint256 unitBalance = deltaBalance /
1654   (denominator - (buyFee.liquidity + sellFee.liquidity));
1655   uint256 bnbToAddLiquidityWith = unitBalance *
1656   (buyFee.liquidity + sellFee.liquidity);
1657
1658
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1654

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1653   uint256 unitBalance = deltaBalance /
1654   (denominator - (buyFee.liquidity + sellFee.liquidity));
1655   uint256 bnbToAddLiquidityWith = unitBalance *
1656   (buyFee.liquidity + sellFee.liquidity);
1657
1658
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1655

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1654   (denominator - (buyFee.liquidity + sellFee.liquidity));
1655   uint256 bnbToAddLiquidityWith = unitBalance *
1656   (buyFee.liquidity + sellFee.liquidity);
1657
1658   if (bnbToAddLiquidityWith > 0) {
1659
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1656

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1655   uint256 bnbToAddLiquidityWith = unitBalance *
1656   (buyFee.liquidity + sellFee.liquidity);
1657
1658   if (bnbToAddLiquidityWith > 0) {
1659   // Add liquidity to pancake
1660
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1664

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1663   // Send ETH to marketing
1664   uint256 marketingAmt = unitBalance *
1665   2 *
1666   (buyFee.marketing + sellFee.marketing);
1667   uint256 teamAmt = unitBalance * 2 * (buyFee.team + sellFee.team) >
1668
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1664

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1663   // Send ETH to marketing
1664   uint256 marketingAmt = unitBalance *
1665   2 *
1666   (buyFee.marketing + sellFee.marketing);
1667   uint256 teamAmt = unitBalance * 2 * (buyFee.team + sellFee.team) >
1668
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1666

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1665   2 *
1666   (buyFee.marketing + sellFee.marketing);
1667   uint256 teamAmt = unitBalance * 2 * (buyFee.team + sellFee.team) >
1668   address(this).balance
1669   ? address(this).balance
1670
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1667

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1666   (buyFee.marketing + sellFee.marketing);
1667   uint256 teamAmt = unitBalance * 2 * (buyFee.team + sellFee.team) >
1668   address(this).balance
1669   ? address(this).balance
1670   : unitBalance * 2 * (buyFee.team + sellFee.team);
1671
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1667

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1666   (buyFee.marketing + sellFee.marketing);
1667   uint256 teamAmt = unitBalance * 2 * (buyFee.team + sellFee.team) >
1668   address(this).balance
1669   ? address(this).balance
1670   : unitBalance * 2 * (buyFee.team + sellFee.team);
1671
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1667

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1666   (buyFee.marketing + sellFee.marketing);
1667   uint256 teamAmt = unitBalance * 2 * (buyFee.team + sellFee.team) >
1668   address(this).balance
1669   ? address(this).balance
1670   : unitBalance * 2 * (buyFee.team + sellFee.team);
1671
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1670

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1669   ? address(this).balance
1670   : unitBalance * 2 * (buyFee.team + sellFee.team);
1671
1672   if (marketingAmt > 0) {
1673   payable(_marketingAddress).transfer(marketingAmt);
1674
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1670

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1669   ? address(this).balance
1670   : unitBalance * 2 * (buyFee.team + sellFee.team);
1671
1672   if (marketingAmt > 0) {
1673   payable(_marketingAddress).transfer(marketingAmt);
1674
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1670

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1669   ? address(this).balance
1670   : unitBalance * 2 * (buyFee.team + sellFee.team);
1671
1672   if (marketingAmt > 0) {
1673   payable(_marketingAddress).transfer(marketingAmt);
1674
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 1253

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1252   _blackListedBots[i] = _blackListedBots[
1253   _blackListedBots.length - 1
1254   ];
1255   _isBlackListedBot[account] = false;
1256   _blackListedBots.pop();
1257
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 1275

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1274   if (_excluded[i] == account) {
1275   _excluded[i] = _excluded[_excluded.length - 1];
1276   _tOwned[account] = 0;
1277   _isExcluded[account] = false;
1278   _excluded.pop();
1279
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 18

low SEVERITY

The current pragma Solidity directive is ""^0.8.10"". 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

- VisionaryDeFi.sol

Locations

```
17
18   pragma solidity ^0.8.10;
19
20   // SPDX-License-Identifier: Unlicensed
21   interface IERC20 {
22
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 936

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
935   payable(address(0x8246A62f2694003fe5120063Ff3D13713316bC76));
936   address _partnershipswallet =
937   payable(address(0xC19752B3C1AC219F865E66C83A52aa5d93f251Ed));
938   address private _burnAddress =
939   0x000000000000000000000000000000000000dEaD;
940
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 973

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
972
973   bool inSwapAndLiquify;
974   bool public swapAndLiquifyEnabled = true;
975
976   uint256 public _maxTxAmount = 12000 * 10**19 * 10**9;
977
```

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

LINE 1580

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- VisionaryDeFi.sol

Locations

```
1579   require(!_isBlackListedBot[msg.sender], "blacklisted");
1580   require(!_isBlackListedBot[tx.origin], "blacklisted");
1581
1582   // is the token balance of this contract address over the min number of
1583   // tokens that we need to initiate a swap + liquidity lock?
1584
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1251

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1250   for (uint256 i = 0; i < _blackListedBots.length; i++) {
1251   if (_blackListedBots[i] == account) {
1252   _blackListedBots[i] = _blackListedBots[
1253   _blackListedBots.length - 1
1254   ];
1255
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1252

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1251   if (_blackListedBots[i] == account) {
1252   _blackListedBots[i] = _blackListedBots[
1253   _blackListedBots.length - 1
1254   ];
1255   _isBlackListedBot[account] = false;
1256
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1252

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1251   if (_blackListedBots[i] == account) {
1252   _blackListedBots[i] = _blackListedBots[
1253   _blackListedBots.length - 1
1254   ];
1255   _isBlackListedBot[account] = false;
1256
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1274

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1273   for (uint256 i = 0; i < _excluded.length; i++) {
1274   if (_excluded[i] == account) {
1275   _excluded[i] = _excluded[_excluded.length - 1];
1276   _tOwned[account] = 0;
1277   _isExcluded[account] = false;
1278
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1275

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1274   if (_excluded[i] == account) {
1275   _excluded[i] = _excluded[_excluded.length - 1];
1276   _tOwned[account] = 0;
1277   _isExcluded[account] = false;
1278   _excluded.pop();
1279
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1275

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1274   if (_excluded[i] == account) {
1275   _excluded[i] = _excluded[_excluded.length - 1];
1276   _tOwned[account] = 0;
1277   _isExcluded[account] = false;
1278   _excluded.pop();
1279
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1286

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1285   for(uint256 i = 0; i < botsWallet.length; i++){
1286   address wallet = botsWallet[i];
1287   uint256 amount = balanceOf(wallet);
1288   _transferStandard(wallet,
1289   address(0x000000000000000000000000000000000000dEaD), amount);
1290
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1458

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1457   if (
1458   _rOwned[_excluded[i]] > rSupply ||
1459   _tOwned[_excluded[i]] > tSupply
1460   ) return (_rTotal, _tTotal);
1461   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1462
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1459

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1458   _rOwned[_excluded[i]] > rSupply ||
1459   _tOwned[_excluded[i]] > tSupply
1460   ) return (_rTotal, _tTotal);
1461   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1462   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1463
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1461

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1460   ) return (_rTotal, _tTotal);
1461   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1462   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1463   }
1464   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1465
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1462

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1461   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1462   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1463   }
1464   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1465   return (rSupply, tSupply);
1466
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1684

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1683   address[] memory path = new address[](2);
1684   path[0] = address(this);
1685   path[1] = uniswapV2Router.WETH();
1686
1687   _approve(address(this), address(uniswapV2Router), tokenAmount);
1688
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1685

low SEVERITY

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

Source File

- VisionaryDeFi.sol

Locations

```
1684   path[0] = address(this);
1685   path[1] = uniswapV2Router.WETH();
1686
1687   _approve(address(this), address(uniswapV2Router), tokenAmount);
1688
1689
```

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.