NanoMatic

Smart Contract Audit Report

SYSFIXED

11 Mar 2023

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
NanoMatic	NANO	Binance Smart Chain

Addresses

Contract address	0xb15488af39bd1de209d501672a293bcd05f82ab4
Contract deployer address	0x6A8160398eb57cb4d0C74d7B5C0F74b4a2D29b07

Project Website

https://www.nanomatic.io/

Codebase

https://bscscan.com/address/0xb15488af39bd1de209d501672a293bcd05f82ab4#code

SUMMARY

NanoMatic is a deflationary Matic rewards token on the Binance Smart Chain. NanoMatic embellishes a state-of-the-art rewards distributor, offering 10% Matic Rewards on both buys and sells. The token will serve as the form of currency of OptDex, a revolutionary DeFi Cryptocurrency Options trading platform slated to be released in late 2023. NanoMatic will launch with an initial supply of 10 million, a “Nano” amount compared to Matic’s monstrous 10 billion result supply. The token, however, will be deflationary as NanoMatic will offer a revolutionary concept of Burn & Sync.

Contract Summary

Documentation Quality

NanoMatic 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 NanoMatic 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 849, 943, 944 and 946.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 105, 137, 160, 161, 196, 232, 259, 263, 275, 282, 291, 375, 665, 865, 952, 952, 1003, 1003, 1016, 1019, 1172, 1174, 1216, 1216, 1222, 1229, 1299, 1318, 1323, 1323, 1323, 1323, 1323, 1324, 1385, 1411, 1411, 1450, 1450, 1450, 1450, 1450, 1474, 1480, 1499, 1508, 1708, 1757, 1779, 1787, 1956, 1966, 1969, 375 and 1174.
- 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 346, 376, 381, 1173, 1174, 1174, 1300, 1300, 1301, 1302, 1549, 1550, 1567, 1568, 1581, 1582, 1583 and 1962.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 1424 and 2007.

CONCLUSION

We have audited the NanoMatic project released on March 2021 to discover issues and identify potential security vulnerabilities in NanoMatic 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 satisfactory results with low-risk issues.

The issues found in the NanoMatic 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 arithmetic operation issues, a state variable visibility is not set, tx.origin as a part of authorization control, and out-of-bounds array access in which the index access expression can cause an exception to the use of an invalid array index value. State variable visibility is not set, the best practice is to set the visibility of state variables explicitly. The default visibility for "walletFeeInBNB" is internal. Other possible visibility settings are public and private. Use of "tx.origin" as a part of authorization control. 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.

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.	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.	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	Friday Mar 10 2023 11:09:00 GMT+0000 (Coordinated Universal Time)
Finished	Saturday Mar 11 2023 08:05:13 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Token.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged

SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	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-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 105

low SEVERITY

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

Source File

- Token.sol

Locations

```
104   function add(uint256 a, uint256 b) internal pure returns (uint256) {
105   uint256 c = a + b;
106   require(c >= a, "SafeMath: addition overflow");
107
108   return c;
109
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 137

low SEVERITY

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

Source File

- Token.sol

Locations

```
136   require(b <= a, errorMessage);
137   uint256 c = a - b;
138
139   return c;
140   }
141
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 160

low SEVERITY

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

Source File

- Token.sol

Locations

```
159
160   uint256 c = a * b;
161   require(c / a == b, "SafeMath: multiplication overflow");
162
163   return c;
164
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 161

low SEVERITY

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

Source File

- Token.sol

Locations

```
160   uint256 c = a * b;
161   require(c / a == b, "SafeMath: multiplication overflow");
162
163   return c;
164   }
165
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 196

low SEVERITY

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

Source File

- Token.sol

Locations

```
195   require(b > 0, errorMessage);
196   uint256 c = a / b;
197   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
198
199   return c;
200
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 232

low SEVERITY

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

Source File

- Token.sol

Locations

```
231   require(b != 0, errorMessage);
232   return a % b;
233   }
234   }
235
236
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 259

low SEVERITY

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

Source File

- Token.sol

Locations

```
258   function mul(int256 a, int256 b) internal pure returns (int256) {
259   int256 c = a * b;
260
261   // Detect overflow when multiplying MIN_INT256 with -1
262   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
263
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 263

low SEVERITY

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

Source File

- Token.sol

Locations

```
262   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
263   require((b == 0) || (c / b == a));
264   return c;
265   }
266
267
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 275

low SEVERITY

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

Source File

- Token.sol

Locations

```
274   // Solidity already throws when dividing by 0.
275   return a / b;
276   }
277
278   /**
279
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 282

low SEVERITY

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

Source File

- Token.sol

Locations

```
281   function sub(int256 a, int256 b) internal pure returns (int256) {
282   int256 c = a - b;
283   require((b >= 0 && c <= a) || (b < 0 && c > a));
284   return c;
285   }
286
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 291

low SEVERITY

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

Source File

- Token.sol

Locations

```
290   function add(int256 a, int256 b) internal pure returns (int256) {
291   int256 c = a + b;
292   require((b >= 0 && c >= a) || (b < 0 && c < a));
293   return c;
294   }
295
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 375

low SEVERITY

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

Source File

- Token.sol

Locations

```
374   uint index = map.indexOf[key];
375   uint lastIndex = map.keys.length - 1;
376   address lastKey = map.keys[lastIndex];
377
378   map.indexOf[lastKey] = index;
379
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 665

low SEVERITY

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

Source File

- Token.sol

Locations

```
664   _owner = address(0);
665   _lockTime = block.timestamp + time;
666   emit OwnershipTransferred(_owner, address(0));
667   }
668
669
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 865

low SEVERITY

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

Source File

- Token.sol

Locations

```
864   uint256 public _tDividendTotal = 0;
865   uint256 internal constant magnitude = 2**128;
866   uint256 internal magnifiedDividendPerShare;
867   mapping(address => int256) internal magnifiedDividendCorrections;
868   mapping(address => uint256) internal withdrawnDividends;
869
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 952

low SEVERITY

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

Source File

- Token.sol

Locations

```
951   uint256 public numTokensSellToAddToLiquidity;
952   uint256 private buyBackUpperLimit = 1 * 10**18;
953
954   mapping(address => bool) public _isBlacklisted;
955
956
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 952

low SEVERITY

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

Source File

- Token.sol

Locations

```
951   uint256 public numTokensSellToAddToLiquidity;
952   uint256 private buyBackUpperLimit = 1 * 10**18;
953
954   mapping(address => bool) public _isBlacklisted;
955
956
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1003

low SEVERITY

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

Source File

- Token.sol

Locations

```
1002   _tTotal = amountOfTokenWei;
1003   _rTotal = (MAX - (MAX % _tTotal));
1004
1005   _rOwned[_msgSender()] = _rTotal;
1006
1007
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1003

low SEVERITY

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

Source File

- Token.sol

Locations

```
1002   _tTotal = amountOfTokenWei;
1003   _rTotal = (MAX - (MAX % _tTotal));
1004
1005   _rOwned[_msgSender()] = _rTotal;
1006
1007
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1016

low SEVERITY

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

Source File

- Token.sol

Locations

```
1015   _maxTxAmount = _tTotal.mul(setMxTxPer).div(
1016   10**4
1017   );
1018   _maxWalletAmount = _tTotal.mul(setMxWalletPer).div(
1019   10**4
1020
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1019

low SEVERITY

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

Source File

- Token.sol

Locations

```
1018   _maxWalletAmount = _tTotal.mul(setMxWalletPer).div(
1019   10**4
1020   );
1021
1022   numTokensSellToAddToLiquidity = amountOfTokenWei.mul(1).div(1000);
1023
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1172

low SEVERITY

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

Source File

- Token.sol

Locations

```
1171   require(_isExcluded[account], "Already excluded");
1172   for (uint256 i = 0; i < _excluded.length; i++) {
1173   if (_excluded[i] == account) {
1174   _excluded[i] = _excluded[_excluded.length - 1];
1175   _tOwned[account] = 0;
1176
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1174

low SEVERITY

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

Source File

- Token.sol

Locations

```
1173   if (_excluded[i] == account) {
1174   _excluded[i] = _excluded[_excluded.length - 1];
1175   _tOwned[account] = 0;
1176   _isExcluded[account] = false;
1177   _excluded.pop();
1178
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1216

low SEVERITY

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

Source File

- Token.sol

Locations

```
1215   function setBuybackUpperLimit(uint256 buyBackLimit) external onlyOwner() {
1216   buyBackUpperLimit = buyBackLimit * 10**18;
1217   }
1218
1219   function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner() {
1220
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1216

low SEVERITY

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

Source File

- Token.sol

Locations

```
1215   function setBuybackUpperLimit(uint256 buyBackLimit) external onlyOwner() {
1216   buyBackUpperLimit = buyBackLimit * 10**18;
1217   }
1218
1219   function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner() {
1220
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1222

low SEVERITY

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

Source File

- Token.sol

Locations

```
1221   _maxTxAmount = _tTotal.mul(maxTxPercent).div(
1222   10**4
1223   );
1224   }
1225
1226
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1229

low SEVERITY

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

Source File

- Token.sol

Locations

```
1228   _maxWalletAmount = _tTotal.mul(maxWalletPercent).div(
1229   10**4
1230   );
1231   }
1232
1233
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1299

low SEVERITY

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

Source File

- Token.sol

Locations

```
1298   uint256 tSupply = _tTotal;
1299   for (uint256 i = 0; i < _excluded.length; i++) {
1300   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
1301   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1302   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1303
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1318

low SEVERITY

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

Source File

- Token.sol

Locations

```
1317   return _amount.mul(_taxFee).div(
1318   10**2
1319   );
1320   }
1321
1322
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1323

low SEVERITY

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

Source File

- Token.sol

Locations

```
1322   function calculateLiquidityFee(uint256 _amount) private view returns (uint256) {
1323   return _amount.mul(_liquidityFee + _burnFee + _walletFee + _buybackFee +
_walletCharityFee + _rewardFee).div(
1324   10**2
1325   );
1326   }
1327
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1323

low SEVERITY

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

Source File

- Token.sol

Locations

```
1322   function calculateLiquidityFee(uint256 _amount) private view returns (uint256) {
1323   return _amount.mul(_liquidityFee + _burnFee + _walletFee + _buybackFee +
_walletCharityFee + _rewardFee).div(
1324   10**2
1325   );
1326   }
1327
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1323

low SEVERITY

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

Source File

- Token.sol

Locations

```
1322   function calculateLiquidityFee(uint256 _amount) private view returns (uint256) {
1323   return _amount.mul(_liquidityFee + _burnFee + _walletFee + _buybackFee +
_walletCharityFee + _rewardFee).div(
1324   10**2
1325   );
1326   }
1327
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1323

low SEVERITY

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

Source File

- Token.sol

Locations

```
1322   function calculateLiquidityFee(uint256 _amount) private view returns (uint256) {
1323   return _amount.mul(_liquidityFee + _burnFee + _walletFee + _buybackFee +
_walletCharityFee + _rewardFee).div(
1324   10**2
1325   );
1326   }
1327
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1323

low SEVERITY

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

Source File

- Token.sol

Locations

```
1322   function calculateLiquidityFee(uint256 _amount) private view returns (uint256) {
1323   return _amount.mul(_liquidityFee + _burnFee + _walletFee + _buybackFee +
_walletCharityFee + _rewardFee).div(
1324   10**2
1325   );
1326   }
1327
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1324

low SEVERITY

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

Source File

- Token.sol

Locations

```
1323   return _amount.mul(_liquidityFee + _burnFee + _walletFee + _buybackFee +
_walletCharityFee + _rewardFee).div(
1324   10**2
1325   );
1326   }
1327
1328
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1385

low SEVERITY

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

Source File

- Token.sol

Locations

```
1384   uint256 contractBalanceRecepient = balanceOf(to);
1385   require(contractBalanceRecepient + amount <= _maxWalletAmount, "Exceeds maximum
wallet amount");
1386   }
1387   // is the token balance of this contract address over the min number of
1388   // tokens that we need to initiate a swap + liquidity lock?
1389
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1411

low SEVERITY

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

Source File

- Token.sol

Locations

```
1410   uint256 balance = address(this).balance;
1411   if (balance > uint256(1 * 10**18)) {
1412
1413   if (balance > buyBackUpperLimit)
1414   balance = buyBackUpperLimit;
1415
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1411

low SEVERITY

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

Source File

- Token.sol

Locations

```
1410   uint256 balance = address(this).balance;
1411   if (balance > uint256(1 * 10**18)) {
1412
1413   if (balance > buyBackUpperLimit)
1414   balance = buyBackUpperLimit;
1415
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1450

low SEVERITY

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

Source File

- Token.sol

Locations

```
1449   //burn
1450   uint8 totFee  = _burnFee + _walletFee + _liquidityFee + _buybackFee +
_walletCharityFee + _rewardFee;
1451   uint256 spentAmount = 0;
1452   uint256 totSpentAmount = 0;
1453   if(_burnFee != 0){
1454
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1450

low SEVERITY

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

Source File

- Token.sol

Locations

```
1449   //burn
1450   uint8 totFee  = _burnFee + _walletFee + _liquidityFee + _buybackFee +
_walletCharityFee + _rewardFee;
1451   uint256 spentAmount = 0;
1452   uint256 totSpentAmount = 0;
1453   if(_burnFee != 0){
1454
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1450

low SEVERITY

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

Source File

- Token.sol

Locations

```
1449   //burn
1450   uint8 totFee  = _burnFee + _walletFee + _liquidityFee + _buybackFee +
_walletCharityFee + _rewardFee;
1451   uint256 spentAmount = 0;
1452   uint256 totSpentAmount = 0;
1453   if(_burnFee != 0){
1454
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1450

low SEVERITY

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

Source File

- Token.sol

Locations

```
1449   //burn
1450   uint8 totFee  = _burnFee + _walletFee + _liquidityFee + _buybackFee +
_walletCharityFee + _rewardFee;
1451   uint256 spentAmount = 0;
1452   uint256 totSpentAmount = 0;
1453   if(_burnFee != 0){
1454
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1450

low SEVERITY

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

Source File

- Token.sol

Locations

```
1449   //burn
1450   uint8 totFee  = _burnFee + _walletFee + _liquidityFee + _buybackFee +
_walletCharityFee + _rewardFee;
1451   uint256 spentAmount = 0;
1452   uint256 totSpentAmount = 0;
1453   if(_burnFee != 0){
1454
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1474

low SEVERITY

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

Source File

- Token.sol

Locations

```
1473   }
1474   totSpentAmount = totSpentAmount + spentAmount;
1475   }
1476
1477   if(_buybackFee != 0){
1478
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1480

low SEVERITY

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

Source File

- Token.sol

Locations

```
1479   swapTokensForBNB(spentAmount);
1480   totSpentAmount = totSpentAmount + spentAmount;
1481   }
1482
1483   if(_walletCharityFee != 0){
1484
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1499

low SEVERITY

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

Source File

- Token.sol

Locations

```
1498   }
1499   totSpentAmount = totSpentAmount + spentAmount;
1500   }
1501
1502   if(_rewardFee != 0){
1503
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1508

low SEVERITY

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

Source File

- Token.sol

Locations

```
1507   distributeDividends(newBalance);
1508   totSpentAmount = totSpentAmount + spentAmount;
1509   }
1510
1511   if(_liquidityFee != 0){
1512
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1708

low SEVERITY

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

Source File

- Token.sol

Locations

```
1707   magnifiedDividendPerShare = magnifiedDividendPerShare.add(
1708   (amount).mul(magnitude) / _tDividendTotal
1709   );
1710   emit DividendsDistributed(amount);
1711
1712
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1757

low SEVERITY

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

Source File

- Token.sol

Locations

```
1756   return
1757   magnifiedDividendPerShare
1758   .mul(balanceOf(_owner))
1759   .toInt256Safe()
1760   .add(magnifiedDividendCorrections[_owner])
1761
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1779

low SEVERITY

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

Source File

- Token.sol

Locations

```
1778   function _dmint(address account, uint256 value) internal {
1779   _tDividendTotal = _tDividendTotal + value;
1780   magnifiedDividendCorrections[account] = magnifiedDividendCorrections[account].sub(
1781   (magnifiedDividendPerShare.mul(value)).toInt256Safe()
1782   );
1783
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1787

low SEVERITY

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

Source File

- Token.sol

Locations

```
1786   function _dburn(address account, uint256 value) internal {
1787   _tDividendTotal = _tDividendTotal - value;
1788   magnifiedDividendCorrections[account] = magnifiedDividendCorrections[account].add(
1789   (magnifiedDividendPerShare.mul(value)).toInt256Safe()
1790   );
1791
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1956

low SEVERITY

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

Source File

- Token.sol

Locations

```
1955   while (gasUsed < gas && iterations < numberOfTokenHolders) {
1956   _lastProcessedIndex++;
1957
1958   if (_lastProcessedIndex >= tokenHoldersMap.keys.length) {
1959   _lastProcessedIndex = 0;
1960
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1966

low SEVERITY

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

Source File

- Token.sol

Locations

```
1965   if (processAccount(payable(account), true)) {
1966   claims++;
1967   }
1968   }
1969   iterations++;
1970
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1969

low SEVERITY

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

Source File

- Token.sol

Locations

```
1968   }
1969   iterations++;
1970   uint256 newGasLeft = gasleft();
1971   if (gasLeft > newGasLeft) {
1972   gasUsed = gasUsed.add(gasLeft.sub(newGasLeft));
1973
```

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

LINE 375

low SEVERITY

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

Source File

- Token.sol

Locations

```
374   uint index = map.indexOf[key];
375   uint lastIndex = map.keys.length - 1;
376   address lastKey = map.keys[lastIndex];
377
378   map.indexOf[lastKey] = index;
379
```

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

LINE 1174

low SEVERITY

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

Source File

- Token.sol

Locations

```
1173   if (_excluded[i] == account) {
1174   _excluded[i] = _excluded[_excluded.length - 1];
1175   _tOwned[account] = 0;
1176   _isExcluded[account] = false;
1177   _excluded.pop();
1178
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 849

low SEVERITY

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

Source File

- Token.sol

Locations

```
848
849   address dead = 0x000000000000000000000000000000000000dEaD;
850
851   uint8 public maxLiqFee = 10;
852   uint8 public maxTaxFee = 10;
853
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 943

low SEVERITY

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

Source File

- Token.sol

Locations

```
942
943   bool walletFeeInBNB = false;
944   bool walletCharityFeeInBNB = false;
945
946   bool inSwapAndLiquify;
947
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 944

low SEVERITY

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

Source File

- Token.sol

Locations

```
943   bool walletFeeInBNB = false;
944   bool walletCharityFeeInBNB = false;
945
946   bool inSwapAndLiquify;
947   bool public swapAndLiquifyEnabled = true;
948
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 946

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

- Token.sol

Locations

```
945
946   bool inSwapAndLiquify;
947   bool public swapAndLiquifyEnabled = true;
948
949   uint256 public _maxTxAmount;
950
```

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

LINE 1424

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

- Token.sol

Locations

```
1423   (uint256 iterations, uint256 claims, uint256 _lastProcessedIndex) = process(gas);
1424   emit ProcessedDividendTracker(iterations, claims, _lastProcessedIndex, true, gas,
tx.origin);
1425   }
1426   }
1427
1428
```

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

LINE 2007

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

- Token.sol

Locations

```
2006   (uint256 iterations, uint256 claims, uint256 _lastProcessedIndex) = process(gas);
2007   emit ProcessedDividendTracker(iterations, claims, _lastProcessedIndex, false, gas,
tx.origin);
2008   }
2009
2010   function blacklistAddress(address account, bool value) external onlyOwner {
2011
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 346

low SEVERITY

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

Source File

- Token.sol

Locations

```
345   function getKeyAtIndex(Map storage map, uint index) internal view returns (address)
{
346   return map.keys[index];
347   }
348
349
350
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 376

low SEVERITY

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

Source File

- Token.sol

Locations

```
375   uint lastIndex = map.keys.length - 1;
376   address lastKey = map.keys[lastIndex];
377
378   map.indexOf[lastKey] = index;
379   delete map.indexOf[key];
380
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 381

low SEVERITY

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

Source File

- Token.sol

Locations

```
380
381   map.keys[index] = lastKey;
382   map.keys.pop();
383   }
384   }
385
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1173

low SEVERITY

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

Source File

- Token.sol

Locations

```
1172   for (uint256 i = 0; i < _excluded.length; i++) {
1173   if (_excluded[i] == account) {
1174   _excluded[i] = _excluded[_excluded.length - 1];
1175   _tOwned[account] = 0;
1176   _isExcluded[account] = false;
1177
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1174

low SEVERITY

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

Source File

- Token.sol

Locations

```
1173   if (_excluded[i] == account) {
1174   _excluded[i] = _excluded[_excluded.length - 1];
1175   _tOwned[account] = 0;
1176   _isExcluded[account] = false;
1177   _excluded.pop();
1178
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1174

low SEVERITY

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

Source File

- Token.sol

Locations

```
1173   if (_excluded[i] == account) {
1174   _excluded[i] = _excluded[_excluded.length - 1];
1175   _tOwned[account] = 0;
1176   _isExcluded[account] = false;
1177   _excluded.pop();
1178
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1300

low SEVERITY

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

Source File

- Token.sol

Locations

```
1299   for (uint256 i = 0; i < _excluded.length; i++) {
1300   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
1301   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1302   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1303   }
1304
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1300

low SEVERITY

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

Source File

- Token.sol

Locations

```
1299   for (uint256 i = 0; i < _excluded.length; i++) {
1300   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
1301   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1302   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1303   }
1304
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1301

low SEVERITY

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

Source File

- Token.sol

Locations

```
1300   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
1301   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1302   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1303   }
1304   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1305
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1302

low SEVERITY

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

Source File

- Token.sol

Locations

```
1301   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1302   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1303   }
1304   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1305   return (rSupply, tSupply);
1306
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1549

low SEVERITY

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

Source File

- Token.sol

Locations

```
1548   address[] memory path = new address[](2);
1549   path[0] = address(this);
1550   path[1] = pcsV2Router.WETH();
1551
1552   _approve(address(this), address(pcsV2Router), tokenAmount);
1553
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1550

low SEVERITY

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

Source File

- Token.sol

Locations

```
1549   path[0] = address(this);
1550   path[1] = pcsV2Router.WETH();
1551
1552   _approve(address(this), address(pcsV2Router), tokenAmount);
1553
1554
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1567

low SEVERITY

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

Source File

- Token.sol

Locations

```
1566   address[] memory path = new address[](2);
1567   path[0] = pcsV2Router.WETH();
1568   path[1] = address(this);
1569
1570   // make the swap
1571
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1568

low SEVERITY

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

Source File

- Token.sol

Locations

```
1567   path[0] = pcsV2Router.WETH();
1568   path[1] = address(this);
1569
1570   // make the swap
1571   pcsV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(
1572
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1581

low SEVERITY

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

Source File

- Token.sol

Locations

```
1580   address[] memory path = new address[](3);
1581   path[0] = address(this);
1582   path[1] = pcsV2Router.WETH();
1583   path[2] = rewardToken;
1584
1585
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1582

low SEVERITY

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

Source File

- Token.sol

Locations

```
1581   path[0] = address(this);
1582   path[1] = pcsV2Router.WETH();
1583   path[2] = rewardToken;
1584
1585   _approve(address(this), address(pcsV2Router), tokenAmount);
1586
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1583

low SEVERITY

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

Source File

- Token.sol

Locations

```
1582   path[1] = pcsV2Router.WETH();
1583   path[2] = rewardToken;
1584
1585   _approve(address(this), address(pcsV2Router), tokenAmount);
1586
1587
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1962

low SEVERITY

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

Source File

- Token.sol

Locations

```
1961
1962   address account = tokenHoldersMap.keys[_lastProcessedIndex];
1963
1964   if (canAutoClaim(lastClaimTimes[account])) {
1965   if (processAccount(payable(account), true)) {
1966
```

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.