WET Token

Smart Contract Audit Report

SYSFIXED

13 Dec 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
WET Token	WET	Binance Smart Chain

Addresses

Contract address	0x324Ca33Dc70Ce3010AA70c1F94940Dd5C133490F
Contract deployer address	0x5066723eDf8af6455c9d1099C047e1EaBfB46b3b

Project Website

https://wethub.co/

Codebase

https://bscscan.com/address/0x324Ca33Dc70Ce3010AA70c1F94940Dd5C133490F#code

SUMMARY

WetHub is a Web 3.0 social networking platform for content creators, that helps them earn extra income from donations, and subscriptions from followers, and fans who are crypto users. NO presale, NO private sale, Audited by BlockSafu, a company recommended by PinkSale, Lock in liquidity for 1 year, Release 100% tokens immediately after listing on PancakeSwap, NFTs System, V1 Platform live now. Discount 10% for max contribution (2 BNB)

Contract Summary

Documentation Quality

WET Token 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 WET Token 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 955.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 209, 223, 238, 239, 252, 264, 279, 293, 307, 321, 337, 360, 383, 409, 923, 991, 991, 1000, 1000, 1012, 1194, 1196, 1236, 1236, 1247, 1247, 1255, 1255, 1262, 1366, 1400, 1408, 1417 and 1196.
- 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 1195, 1196, 1196, 1368, 1369, 1371, 1372, 1518 and 1519.

CONCLUSION

We have audited the WET Token project released on December 2022 to discover issues and identify potential security vulnerabilities in WET Token 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 WET Token smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a state variable visibility is not set, 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.	PASS
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

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

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

SMART CONTRACT ANALYSIS

Started	Monday Dec 12 2022 15:49:40 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Dec 13 2022 02:30:32 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	LiquidityGeneratorToken.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	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 209

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
208   unchecked {
209   uint256 c = a + b;
210   if (c < a) return (false, 0);
211   return (true, c);
212   }
213
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 223

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
222   if (b > a) return (false, 0);
223   return (true, a - b);
224   }
225   }
226
227
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 238

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
237   if (a == 0) return (true, 0);
238   uint256 c = a * b;
239   if (c / a != b) return (false, 0);
240   return (true, c);
241   }
242
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 239

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
238   uint256 c = a * b;
239   if (c / a != b) return (false, 0);
240   return (true, c);
241   }
242   }
243
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 252

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
251   if (b == 0) return (false, 0);
252   return (true, a / b);
253   }
254   }
255
256
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 264

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
263   if (b == 0) return (false, 0);
264   return (true, a % b);
265   }
266   }
267
268
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 279

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
278   function add(uint256 a, uint256 b) internal pure returns (uint256) {
279   return a + b;
280   }
281
282   /**
283
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 293

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
292   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
293   return a - b;
294   }
295
296   /**
297
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 307

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
306   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
307   return a * b;
308   }
309
310   /**
311
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 321

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
320   function div(uint256 a, uint256 b) internal pure returns (uint256) {
321   return a / b;
322   }
323
324   /**
325
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 337

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
336   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
337   return a % b;
338   }
339
340   /**
341
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 360

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
359   require(b <= a, errorMessage);
360   return a - b;
361   }
362   }
363
364
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 383

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
382   require(b > 0, errorMessage);
383   return a / b;
384   }
385   }
386
387
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 409

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
408   require(b > 0, errorMessage);
409   return a % b;
410   }
411   }
412   }
413
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 923

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
922
923   uint256 public constant MAX_FEE = 10**3;
924
925   mapping(address => uint256) private _rOwned;
926   mapping(address => uint256) private _tOwned;
927
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 991

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
990   require(
991   taxFeeBps_ + liquidityFeeBps_ + marketingFeeBps_ <= MAX_FEE,
992   "Total fee is over 10%"
993   );
994
995
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 991

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
990   require(
991   taxFeeBps_ + liquidityFeeBps_ + marketingFeeBps_ <= MAX_FEE,
992   "Total fee is over 10%"
993   );
994
995
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1000

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
999   _tTotal = totalSupply_;
1000   _rTotal = (MAX - (MAX % _tTotal));
1001
1002   _taxFee = taxFeeBps_;
1003   _previousTaxFee = _taxFee;
1004
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1000

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
999   _tTotal = totalSupply_;
1000   _rTotal = (MAX - (MAX % _tTotal));
1001
1002   _taxFee = taxFeeBps_;
1003   _previousTaxFee = _taxFee;
1004
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1012

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1011
1012   numTokensSellToAddToLiquidity = totalSupply_.div(10**3); // 0.1%
1013
1014   swapAndLiquifyEnabled = true;
1015
1016
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1194

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1193   require(_isExcluded[account], "Account is already excluded");
1194   for (uint256 i = 0; i < _excluded.length; i++) {
1195   if (_excluded[i] == account) {
1196   _excluded[i] = _excluded[_excluded.length - 1];
1197   _tOwned[account] = 0;
1198
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1196

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1195   if (_excluded[i] == account) {
1196   _excluded[i] = _excluded[_excluded.length - 1];
1197   _tOwned[account] = 0;
1198   _isExcluded[account] = false;
1199   _excluded.pop();
1200
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1236

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1235   require(
1236   _taxFee + _liquidityFee + _marketingFee <= MAX_FEE,
1237   "Total fee is over 10%"
1238   );
1239   }
1240
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1236

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1235   require(
1236   _taxFee + _liquidityFee + _marketingFee <= MAX_FEE,
1237   "Total fee is over 10%"
1238   );
1239   }
1240
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1247

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1246   require(
1247   _taxFee + _liquidityFee + _marketingFee <= MAX_FEE,
1248   "Total fee is over 10%"
1249   );
1250   }
1251
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1247

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1246   require(
1247   _taxFee + _liquidityFee + _marketingFee <= MAX_FEE,
1248   "Total fee is over 10%"
1249   );
1250   }
1251
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1255

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1254   require(
1255   _taxFee + _liquidityFee + _marketingFee <= MAX_FEE,
1256   "Total fee is over 10%"
1257   );
1258   }
1259
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1255

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1254   require(
1255   _taxFee + _liquidityFee + _marketingFee <= MAX_FEE,
1256   "Total fee is over 10%"
1257   );
1258   }
1259
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1262

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1261   require(
1262   _amount >= totalSupply().mul(5).div(10**4),
1263   "Swapback amount should be at least 0.05% of total supply"
1264   );
1265   numTokensSellToAddToLiquidity = _amount;
1266
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1366

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1365   uint256 tSupply = _tTotal;
1366   for (uint256 i = 0; i < _excluded.length; i++) {
1367   if (
1368   _rOwned[_excluded[i]] > rSupply ||
1369   _tOwned[_excluded[i]] > tSupply
1370
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1400

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1399   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1400   return _amount.mul(_taxFee).div(10**4);
1401   }
1402
1403   function calculateLiquidityFee(uint256 _amount)
1404
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1408

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1407   {
1408   return _amount.mul(_liquidityFee).div(10**4);
1409   }
1410
1411   function calculateMarketingFee(uint256 _amount)
1412
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1417

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1416   if (_marketingAddress == address(0)) return 0;
1417   return _amount.mul(_marketingFee).div(10**4);
1418   }
1419
1420   function removeAllFee() private {
1421
```

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

LINE 1196

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1195   if (_excluded[i] == account) {
1196   _excluded[i] = _excluded[_excluded.length - 1];
1197   _tOwned[account] = 0;
1198   _isExcluded[account] = false;
1199   _excluded.pop();
1200
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 955

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

- LiquidityGeneratorToken.sol

Locations

```
954
955   bool inSwapAndLiquify;
956   bool public swapAndLiquifyEnabled;
957
958   uint256 private numTokensSellToAddToLiquidity;
959
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1195

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1194   for (uint256 i = 0; i < _excluded.length; i++) {
1195   if (_excluded[i] == account) {
1196   _excluded[i] = _excluded[_excluded.length - 1];
1197   _tOwned[account] = 0;
1198   _isExcluded[account] = false;
1199
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1196

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1195   if (_excluded[i] == account) {
1196   _excluded[i] = _excluded[_excluded.length - 1];
1197   _tOwned[account] = 0;
1198   _isExcluded[account] = false;
1199   _excluded.pop();
1200
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1196

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1195   if (_excluded[i] == account) {
1196   _excluded[i] = _excluded[_excluded.length - 1];
1197   _tOwned[account] = 0;
1198   _isExcluded[account] = false;
1199   _excluded.pop();
1200
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1368

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1367   if (
1368   _rOwned[_excluded[i]] > rSupply ||
1369   _tOwned[_excluded[i]] > tSupply
1370   ) return (_rTotal, _tTotal);
1371   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1372
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1369

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1368   _rOwned[_excluded[i]] > rSupply ||
1369   _tOwned[_excluded[i]] > tSupply
1370   ) return (_rTotal, _tTotal);
1371   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1372   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1373
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1371

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1370   ) return (_rTotal, _tTotal);
1371   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1372   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1373   }
1374   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1375
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1372

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1371   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1372   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1373   }
1374   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1375   return (rSupply, tSupply);
1376
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1518

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1517   address[] memory path = new address[](2);
1518   path[0] = address(this);
1519   path[1] = uniswapV2Router.WETH();
1520
1521   _approve(address(this), address(uniswapV2Router), tokenAmount);
1522
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1519

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1518   path[0] = address(this);
1519   path[1] = uniswapV2Router.WETH();
1520
1521   _approve(address(this), address(uniswapV2Router), tokenAmount);
1522
1523
```

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.