Wall Finance

Smart Contract Audit Report

SYSFIXED

14 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Wall Finance	WALL	Binance Smart Chain

Addresses

Contract address	0x33D512a749f6feFaDB832c91c0F23Bc27bE2E3d4
Contract deployer address	0x8A16D5C04E2F123BA35866fedbD5658044e3dD52

Project Website

https://https://wallfinance.net/

Codebase

https://bscscan.com/address/0x33D512a749f6feFaDB832c91c0F23Bc27bE2E3d4#code

SUMMARY

Secured multi-coin wallet that supports Bitcoin, Ethereum, BNB and fully shielded Zcash, as well as other coins, and it has a strong, user-centric architecture in which the users own their own keys and their own privacy. Wall Wallet live on Google Playstore https://play.google.com/store/apps/details?id=io.horizontalsystems wallfinance.

Contract Summary

Documentation Quality

Wall Finance 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 Wall Finance 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 959.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 213, 227, 242, 243, 256, 268, 283, 297, 311, 325, 341, 364, 387, 413, 927, 927, 997, 997, 1006, 1006, 1018, 1202, 1204, 1244, 1244, 1255, 1255, 1263, 1263, 1270, 1374, 1408, 1416, 1425 and 1204.
- 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 1203, 1204, 1204, 1376, 1377, 1379, 1380, 1526 and 1527.

CONCLUSION

We have audited the Wall Finance project released on January 2023 to discover issues and identify potential security vulnerabilities in Wall Finance 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 Wall Finance smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, a 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	Friday Jan 13 2023 21:32:59 GMT+0000 (Coordinated Universal Time)
Finished	Saturday Jan 14 2023 18:52:22 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 213

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
212   unchecked {
213   uint256 c = a + b;
214   if (c < a) return (false, 0);
215   return (true, c);
216   }
217
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 227

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
226   if (b > a) return (false, 0);
227   return (true, a - b);
228   }
229   }
230
231
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 242

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
241   if (a == 0) return (true, 0);
242   uint256 c = a * b;
243   if (c / a != b) return (false, 0);
244   return (true, c);
245   }
246
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 243

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
uint256 c = a * b;
if (c / a != b) return (false, 0);
return (true, c);
}
}

```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 256

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
255   if (b == 0) return (false, 0);
256   return (true, a / b);
257   }
258   }
259
260
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 268

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
267   if (b == 0) return (false, 0);
268   return (true, a % b);
269   }
270   }
271
272
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 283

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
282   function add(uint256 a, uint256 b) internal pure returns (uint256) {
283   return a + b;
284   }
285
286   /**
287
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 297

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
296   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
297   return a - b;
298   }
299
300   /**
301
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 311

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
310   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
311   return a * b;
312   }
313
314   /**
315
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 325

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
function div(uint256 a, uint256 b) internal pure returns (uint256) {
return a / b;
}

/**

```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 341

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
340   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
341   return a % b;
342   }
343
344   /**
345
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 364

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
363   require(b <= a, errorMessage);
364   return a - b;
365   }
366   }
367
368
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 387

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
386   require(b > 0, errorMessage);
387   return a / b;
388   }
389   }
390
391
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 413

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
412   require(b > 0, errorMessage);
413   return a % b;
414   }
415   }
416   }
417
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 927

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
926
927   uint256 public constant MAX_FEE = 10**4 / 4;
928
929   mapping(address => uint256) private _rOwned;
930   mapping(address => uint256) private _tOwned;
931
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 927

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
926
927   uint256 public constant MAX_FEE = 10**4 / 4;
928
929   mapping(address => uint256) private _rOwned;
930   mapping(address => uint256) private _tOwned;
931
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 997

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
996   require(
997   taxFeeBps_ + liquidityFeeBps_ + charityFeeBps_ <= MAX_FEE,
998   "Total fee is over 25%"
999   );
1000
1001
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 997

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
996   require(
997   taxFeeBps_ + liquidityFeeBps_ + charityFeeBps_ <= MAX_FEE,
998   "Total fee is over 25%"
999   );
1000
1001
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1006

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1005   _tTotal = totalSupply_;
1006   _rTotal = (MAX - (MAX % _tTotal));
1007
1008   _taxFee = taxFeeBps_;
1009   _previousTaxFee = _taxFee;
1010
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1006

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1005   _tTotal = totalSupply_;
1006   _rTotal = (MAX - (MAX % _tTotal));
1007
1008   _taxFee = taxFeeBps_;
1009   _previousTaxFee = _taxFee;
1010
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1018

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1017
1018   numTokensSellToAddToLiquidity = totalSupply_.div(10**3); // 0.1%
1019
1020   swapAndLiquifyEnabled = true;
1021
1022
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1202

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1201   require(_isExcluded[account], "Account is already excluded");
1202   for (uint256 i = 0; i < _excluded.length; i++) {
1203   if (_excluded[i] == account) {
1204   _excluded[i] = _excluded[_excluded.length - 1];
1205   _tOwned[account] = 0;
1206
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1204

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1203   if (_excluded[i] == account) {
1204   _excluded[i] = _excluded[_excluded.length - 1];
1205   _tOwned[account] = 0;
1206   _isExcluded[account] = false;
1207   _excluded.pop();
1208
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1244

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
require(
_taxFee + _liquidityFee + _charityFee <= MAX_FEE,
"Total fee is over 25%"
);
}

```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1244

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1243   require(
1244   _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
1245   "Total fee is over 25%"
1246   );
1247   }
1248
```

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 + _charityFee <= MAX_FEE,
1256   "Total fee is over 25%"
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 + _charityFee <= MAX_FEE,
1256   "Total fee is over 25%"
1257   );
1258   }
1259
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1263

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1262   require(
1263   _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
1264   "Total fee is over 25%"
1265   );
1266   }
1267
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1263

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1262   require(
1263   _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
1264   "Total fee is over 25%"
1265   );
1266   }
1267
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1270

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1269   require(
1270   _amount >= totalSupply().mul(5).div(10**4),
1271   "Swapback amount should be at least 0.05% of total supply"
1272   );
1273   numTokensSellToAddToLiquidity = _amount;
1274
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1374

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1373   uint256 tSupply = _tTotal;
1374   for (uint256 i = 0; i < _excluded.length; i++) {
1375   if (
1376   _rOwned[_excluded[i]] > rSupply ||
1377   _tOwned[_excluded[i]] > tSupply
1378
```

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   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1408   return _amount.mul(_taxFee).div(10**4);
1409   }
1410
1411   function calculateLiquidityFee(uint256 _amount)
1412
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1416

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1415   {
1416   return _amount.mul(_liquidityFee).div(10**4);
1417   }
1418
1419   function calculateCharityFee(uint256 _amount)
1420
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1425

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1424   if (_charityAddress == address(0)) return 0;
1425   return _amount.mul(_charityFee).div(10**4);
1426   }
1427
1428   function removeAllFee() private {
1429
```

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

LINE 1204

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1203   if (_excluded[i] == account) {
1204   _excluded[i] = _excluded[_excluded.length - 1];
1205   _tOwned[account] = 0;
1206   _isExcluded[account] = false;
1207   _excluded.pop();
1208
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 959

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

```
958
959   bool inSwapAndLiquify;
960   bool public swapAndLiquifyEnabled;
961
962   uint256 private numTokensSellToAddToLiquidity;
963
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1203

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1202   for (uint256 i = 0; i < _excluded.length; i++) {
1203   if (_excluded[i] == account) {
1204   _excluded[i] = _excluded[_excluded.length - 1];
1205   _tOwned[account] = 0;
1206   _isExcluded[account] = false;
1207
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1204

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1203   if (_excluded[i] == account) {
1204   _excluded[i] = _excluded[_excluded.length - 1];
1205   _tOwned[account] = 0;
1206   _isExcluded[account] = false;
1207   _excluded.pop();
1208
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1204

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1203   if (_excluded[i] == account) {
1204   _excluded[i] = _excluded[_excluded.length - 1];
1205   _tOwned[account] = 0;
1206   _isExcluded[account] = false;
1207   _excluded.pop();
1208
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1376

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1375   if (
1376   _rOwned[_excluded[i]] > rSupply ||
1377   _tOwned[_excluded[i]] > tSupply
1378   ) return (_rTotal, _tTotal);
1379   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1380
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1377

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1376   _rOwned[_excluded[i]] > rSupply ||
1377   _tOwned[_excluded[i]] > tSupply
1378   ) return (_rTotal, _tTotal);
1379   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1380   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1381
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1379

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1378   ) return (_rTotal, _tTotal);
1379   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1380   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1381   }
1382   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1383
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1380

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1379   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1380   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1381   }
1382   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1383   return (rSupply, tSupply);
1384
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1526

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1525   address[] memory path = new address[](2);
1526   path[0] = address(this);
1527   path[1] = uniswapV2Router.WETH();
1528
1529   _approve(address(this), address(uniswapV2Router), tokenAmount);
1530
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1527

low SEVERITY

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

Source File

- LiquidityGeneratorToken.sol

Locations

```
1526    path[0] = address(this);
1527    path[1] = uniswapV2Router.WETH();
1528
1529    _approve(address(this), address(uniswapV2Router), tokenAmount);
1530
1531
```

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.