



AKITSUKI

# Smart Contract Audit Report

# 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
AKITSUKI	AKI	Ethereum

## Addresses

Contract address	0x9eFdFC77017cB9b878300eCdcA21fAB0529a22A0
Contract deployer address	0x1F0D529287B2acaD6480ba771bd73862C9a1b0f5

## Project Website

<https://akitsuki.org/>

## Codebase

<https://etherscan.io/address/0x9eFdFC77017cB9b878300eCdcA21fAB0529a22A0#code>

# SUMMARY

Hello Holy Guards! After a successful Guardian Game on League of Legends, the clan is getting stronger! Many players are joining us! Protectors discord server boosters are receiving a share of the cash prize as well. Something is cooking, stay tuned.

## Contract Summary

### Documentation Quality

AKITSUKI provides a very good documentation with standard of solidity base code.

- The technical description is provided clearly and structured and also don't have any high risk issue.

### Code Quality

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by AKITSUKI 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 957.
- 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, 998, 998, 998, 998, 1007, 1007, 1019, 1203, 1205, 1249, 1249, 1249, 1249, 1260, 1260, 1260, 1260, 1367, 1401, 1409, 1418 and 1205.
- 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 1204, 1205, 1205, 1369, 1370, 1372, 1373, 1521 and 1522.

# CONCLUSION

We have audited the AKITSUKI project released on August 2022 to discover issues and identify potential security vulnerabilities in AKITSUKI 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 AKITSUKI 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)	<b>SWC-113</b> <b>SWC-128</b>	Execution of the code should never be blocked by a specific contract state unless required.	<b>PASS</b>
Race Conditions	<b>SWC-114</b>	Race Conditions and Transactions Order Dependency should not be possible.	<b>PASS</b>
Authorization through tx.origin	<b>SWC-115</b>	tx.origin should not be used for authorization.	<b>PASS</b>
Block values as a proxy for time	<b>SWC-116</b>	Block numbers should not be used for time calculations.	<b>PASS</b>
Signature Unique ID	<b>SWC-117</b> <b>SWC-121</b> <b>SWC-122</b>	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	<b>PASS</b>
Incorrect Constructor Name	<b>SWC-118</b>	Constructors are special functions that are called only once during the contract creation.	<b>PASS</b>
Shadowing State Variable	<b>SWC-119</b>	State variables should not be shadowed.	<b>PASS</b>
Weak Sources of Randomness	<b>SWC-120</b>	Random values should never be generated from Chain Attributes or be predictable.	<b>PASS</b>
Write to Arbitrary Storage Location	<b>SWC-124</b>	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	<b>PASS</b>
Incorrect Inheritance Order	<b>SWC-125</b>	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/.	<b>PASS</b>
Insufficient Gas Griefing	<b>SWC-126</b>	Insufficient gas grieving attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	<b>PASS</b>
Arbitrary Jump Function	<b>SWC-127</b>	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	<b>PASS</b>

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 Aug 05 2022 01:08:57 GMT+0000 (Coordinated Universal Time)
Finished	Saturday Aug 06 2022 08:59:11 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

[illegible]

# 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

```
242     uint256 c = a * b;  
243     if (c / a != b) return (false, 0);  
244     return (true, c);  
245 }  
246 }  
247
```

# 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

```
324     function div(uint256 a, uint256 b) internal pure returns (uint256) {  
325         return a / b;  
326     }  
327  
328     /**  
329
```

# 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 998

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
997     require(  
998     taxFeeBps_ + liquidityFeeBps_ + charityFeeBps_ <= 10**4 / 4,  
999     "Total fee is over 25%"  
1000     );  
1001  
1002
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 998

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
997     require(  
998     taxFeeBps_ + liquidityFeeBps_ + charityFeeBps_ <= 10**4 / 4,  
999     "Total fee is over 25%"  
1000     );  
1001  
1002
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 998

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
997     require(  
998     taxFeeBps_ + liquidityFeeBps_ + charityFeeBps_ <= 10**4 / 4,  
999     "Total fee is over 25%"  
1000     );  
1001  
1002
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 998

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
997     require(  
998     taxFeeBps_ + liquidityFeeBps_ + charityFeeBps_ <= 10**4 / 4,  
999     "Total fee is over 25%"  
1000     );  
1001  
1002
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1007

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1006     _tTotal = totalSupply_;
1007     _rTotal = (MAX - (MAX % _tTotal));
1008
1009     _taxFee = taxFeeBps_;
1010     _previousTaxFee = _taxFee;
1011
```

# SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1007

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1006  _tTotal = totalSupply_;  
1007  _rTotal = (MAX - (MAX % _tTotal));  
1008  
1009  _taxFee = taxFeeBps_;  
1010  _previousTaxFee = _taxFee;  
1011
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 1019

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1018
1019     numTokensSellToAddToLiquidity = totalSupply_.mul(5).div(10**4); // 0.05%
1020
1021     swapAndLiquifyEnabled = true;
1022
1023
```



# SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1203

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1202     require(!_isExcluded[account], "Account is already excluded");
1203     for (uint256 i = 0; i < _excluded.length; i++) {
1204         if (_excluded[i] == account) {
1205             _excluded[i] = _excluded[_excluded.length - 1];
1206             _tOwned[account] = 0;
1207         }
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1205

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1204   if (_excluded[i] == account) {  
1205       _excluded[i] = _excluded[_excluded.length - 1];  
1206       _tOwned[account] = 0;  
1207       _isExcluded[account] = false;  
1208       _excluded.pop();  
1209   }
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1249

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1248     require(  
1249         _taxFee + _liquidityFee + _charityFee <= 10**4 / 4,  
1250         "Total fee is over 25%"  
1251     );  
1252 }  
1253
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1249

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1248     require(  
1249         _taxFee + _liquidityFee + _charityFee <= 10**4 / 4,  
1250         "Total fee is over 25%"  
1251     );  
1252 }  
1253
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1249

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1248     require(  
1249         _taxFee + _liquidityFee + _charityFee <= 10**4 / 4,  
1250         "Total fee is over 25%"  
1251     );  
1252 }  
1253
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 1249

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1248     require(  
1249         _taxFee + _liquidityFee + _charityFee <= 10**4 / 4,  
1250         "Total fee is over 25%"  
1251     );  
1252 }  
1253
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1260

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1259     require(  
1260         _taxFee + _liquidityFee + _charityFee <= 10**4 / 4,  
1261         "Total fee is over 25%"  
1262     );  
1263 }  
1264
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1260

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1259     require(  
1260         _taxFee + _liquidityFee + _charityFee <= 10**4 / 4,  
1261         "Total fee is over 25%"  
1262     );  
1263 }  
1264
```



# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1260

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1259     require(  
1260         _taxFee + _liquidityFee + _charityFee <= 10**4 / 4,  
1261         "Total fee is over 25%"  
1262     );  
1263 }  
1264
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 1260

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1259     require(  
1260         _taxFee + _liquidityFee + _charityFee <= 10**4 / 4,  
1261         "Total fee is over 25%"  
1262     );  
1263 }  
1264
```

# SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1367

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1366     uint256 tSupply = _tTotal;
1367     for (uint256 i = 0; i < _excluded.length; i++) {
1368         if (
1369             _rOwned[_excluded[i]] > rSupply ||
1370             _tOwned[_excluded[i]] > tSupply
1371         ) {
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 1401

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1400 function calculateTaxFee(uint256 _amount) private view returns (uint256) {  
1401     return _amount.mul(_taxFee).div(10**4);  
1402 }  
1403  
1404 function calculateLiquidityFee(uint256 _amount)  
1405
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 1409

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1408 {  
1409     return _amount.mul(_liquidityFee).div(10**4);  
1410 }  
1411  
1412 function calculateCharityFee(uint256 _amount)  
1413
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 1418

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1417     if (_charityAddress == address(0)) return 0;
1418     return _amount.mul(_charityFee).div(10**4);
1419 }
1420
1421 function removeAllFee() private {
1422
```

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

LINE 1205

## low SEVERITY

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

## Source File

- LiquidityGeneratorToken.sol

## Locations

```
1204   if (_excluded[i] == account) {  
1205       _excluded[i] = _excluded[_excluded.length - 1];  
1206       _tOwned[account] = 0;  
1207       _isExcluded[account] = false;  
1208       _excluded.pop();  
1209   }
```

## SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 957

### 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

```
956
957  bool inSwapAndLiquify;
958  bool public swapAndLiquifyEnabled;
959
960  uint256 private numTokensSellToAddToLiquidity;
961
```



## 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   for (uint256 i = 0; i < _excluded.length; i++) {  
1204     if (_excluded[i] == account) {  
1205       _excluded[i] = _excluded[_excluded.length - 1];  
1206       _tOwned[account] = 0;  
1207       _isExcluded[account] = false;  
1208     }
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1205

### low SEVERITY

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

### Source File

- LiquidityGeneratorToken.sol

### Locations

```
1204   if (_excluded[i] == account) {  
1205     _excluded[i] = _excluded[_excluded.length - 1];  
1206     _tOwned[account] = 0;  
1207     _isExcluded[account] = false;  
1208     _excluded.pop();  
1209   }
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1205

### low SEVERITY

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

### Source File

- LiquidityGeneratorToken.sol

### Locations

```
1204   if (_excluded[i] == account) {  
1205     _excluded[i] = _excluded[_excluded.length - 1];  
1206     _tOwned[account] = 0;  
1207     _isExcluded[account] = false;  
1208     _excluded.pop();  
1209   }
```

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

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1370

### low SEVERITY

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

### Source File

- LiquidityGeneratorToken.sol

### Locations

```
1369     _rOwned[_excluded[i]] > rSupply ||  
1370     _tOwned[_excluded[i]] > tSupply  
1371     ) return (_rTotal, _tTotal);  
1372     rSupply = rSupply.sub(_rOwned[_excluded[i]]);  
1373     tSupply = tSupply.sub(_tOwned[_excluded[i]]);  
1374
```

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

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1373

### low SEVERITY

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

### Source File

- LiquidityGeneratorToken.sol

### Locations

```
1372   rSupply = rSupply.sub(_rOwned[_excluded[i]]);  
1373   tSupply = tSupply.sub(_tOwned[_excluded[i]]);  
1374   }  
1375   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);  
1376   return (rSupply, tSupply);  
1377
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1521

### low SEVERITY

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

### Source File

- LiquidityGeneratorToken.sol

### Locations

```
1520     address[] memory path = new address[](2);
1521     path[0] = address(this);
1522     path[1] = uniswapV2Router.WETH();
1523
1524     _approve(address(this), address(uniswapV2Router), tokenAmount);
1525
```



## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1522

### low SEVERITY

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

### Source File

- LiquidityGeneratorToken.sol

### Locations

```
1521 path[0] = address(this);  
1522 path[1] = uniswapV2Router.WETH();  
1523  
1524 _approve(address(this), address(uniswapV2Router), tokenAmount);  
1525  
1526
```

# 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.