Elancing

Smart Contract Audit Report

SYSFIXED

22 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Elancing	ELC	BSC

Addresses

Contract address	0x0FBb8C9f52C354eE8072Fdf314F4cFf6dEBB31c8
Contract deployer address	0xeFc979c8F20bED441DA0206C26AC6cE40144C2aB

Project Website

https://elancing.net/

Codebase

https://bscscan.com/address/0x0FBb8C9f52C354eE8072Fdf314F4cFf6dEBB31c8#code

SUMMARY

Elancing is a revolutionary decentralized marketplace for freelance & outsourcing services. Uses advanced PoW technology for low commission. $ELC holders receive 2% rewards in ETH & 2% BNB dividends from service fees! Excellent audit score | Automated buyback & burn mechanism for price support & accumulation | Beta product ready for use & exploration | Huge marketing & Cex Listings offers!

Contract Summary

Documentation Quality

This project has a standard of documentation.

- Technical description provided.

Code Quality

The quality of the code in this project is up to standard.

- The official Solidity style guide is followed.

Test Scope

Project test coverage is 100% (Via Codebase).

Audit Findings Summary

Issues Found

- SWC-101 | Arithmetic operation issues discovered on lines 55, 67, 77, 78, 89, 101, 110, 114, 122, 125, 130, 199, 569, 587, 629, 689, 840, 850, 854, 948, 977, 978, 1060, 1113, 1115, 1131, 1134, 1139, 1144, 1149, 1151, 1158, 1183, 1191, 1193, 1195, 1202, 1204, 1269, and 1273.
- SWC-101 | Compiler-rewritable " - 1" discovered on lines 199 and 1269.
- SWC-110 | Out of bounds array access issues discovered on lines 172, 200, 205, 846, 1121, 1122, 1228, 1229, 1249, 1250, 1266, 1267, and 1269.
- SWC-115 | Use of "Tx.Origin" as a part of authorization Control on lines 1218 and 1345.

CONCLUSION

We have audited the Elancing project which has released on January 2023 to discover issues and identify potential security vulnerabilities in Elancing Project. This process is used to find technical issues and security loopholes that find some common issues in the code.

The security audit report produced satisfactory results with low-risk issues.

Most issues found were low severity and any critical issue such as High Vulnerability was not found. Except for all other issues that were of negligible importance and mostly referred to coding standards and inefficiencies such as arithmetic operation issues, the use of "tx.origin" as a part of authorization control and out of bounds array access which the index access expression can cause an exception in case of 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.	PASS
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
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Check-Effect Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	PASS
Assert Violation	SWC-110	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 Callerr	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
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
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

SMART CONTRACT ANALYSIS

Started	Sat Jan 21 2023 23:12:46 GMT+0000 (Coordinated Universal Time)
Finished	Sun Jan 22 2023 02:20:50 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	ELC.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	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-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 55

low SEVERITY

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

Source File

- ELC.sol

Locations

```
54   function add(uint256 a, uint256 b) internal pure returns (uint256) {
55   uint256 c = a + b;
56   require(c >= a, "SafeMath: addition overflow");
57
58   return c;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 67

low SEVERITY

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

Source File

- ELC.sol

Locations

```
66   require(b <= a, errorMessage);
67   uint256 c = a - b;
68
69   return c;
70   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 77

low SEVERITY

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

Source File

- ELC.sol

Locations

```
76
77   uint256 c = a * b;
78   require(c / a == b, "SafeMath: multiplication overflow");
79
80   return c;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 78

low SEVERITY

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

Source File

- ELC.sol

Locations

```
77   uint256 c = a * b;
78   require(c / a == b, "SafeMath: multiplication overflow");
79
80   return c;
81   }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 89

low SEVERITY

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

Source File

- ELC.sol

Locations

```
88   require(b > 0, errorMessage);
89   uint256 c = a / b;
90   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
91
92   return c;
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 101

low SEVERITY

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

Source File

- ELC.sol

Locations

```
100   require(b != 0, errorMessage);
101   return a % b;
102   }
103   }
104
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 110

low SEVERITY

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

Source File

- ELC.sol

Locations

```
109   function mul(int256 a, int256 b) internal pure returns (int256) {
110   int256 c = a * b;
111
112   // Detect overflow when multiplying MIN_INT256 with -1
113   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 114

low SEVERITY

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

Source File

- ELC.sol

Locations

```
113   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
114   require((b == 0) || (c / b == a));
115   return c;
116   }
117   function div(int256 a, int256 b) internal pure returns (int256) {
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 122

low SEVERITY

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

Source File

- ELC.sol

Locations

```
121   // Solidity already throws when dividing by 0.
122   return a / b;
123   }
124   function sub(int256 a, int256 b) internal pure returns (int256) {
125   int256 c = a - b;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 125

low SEVERITY

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

Source File

- ELC.sol

Locations

```
124   function sub(int256 a, int256 b) internal pure returns (int256) {
125   int256 c = a - b;
126   require((b >= 0 && c <= a) || (b < 0 && c > a));
127   return c;
128   }
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 130

low SEVERITY

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

Source File

- ELC.sol

Locations

```
129   function add(int256 a, int256 b) internal pure returns (int256) {
130   int256 c = a + b;
131   require((b >= 0 && c >= a) || (b < 0 && c < a));
132   return c;
133   }
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 199

low SEVERITY

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

Source File

- ELC.sol

Locations

```
198   uint index = map.indexOf[key];
199   uint lastIndex = map.keys.length - 1;
200   address lastKey = map.keys[lastIndex];
201
202   map.indexOf[lastKey] = index;
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 569

low SEVERITY

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

Source File

- ELC.sol

Locations

```
568
569   uint256 constant internal magnitude = 2**128;
570   uint256 internal magnifiedDividendPerShare;
571   uint256 public totalDividendsDistributed;
572
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 587

low SEVERITY

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

Source File

- ELC.sol

Locations

```
586   magnifiedDividendPerShare = magnifiedDividendPerShare.add(
587   (amount).mul(magnitude) / totalSupply()
588   );
589   emit DividendsDistributed(msg.sender, amount);
590
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 629

low SEVERITY

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

Source File

- ELC.sol

Locations

```
628   function accumulativeDividendOf(address _owner) public view override
returns(uint256) {
629   return magnifiedDividendPerShare.mul(balanceOf(_owner)).toInt256Safe()
630   .add(magnifiedDividendCorrections[_owner]).toUint256Safe() / magnitude;
631   }
632
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 689

low SEVERITY

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

Source File

- ELC.sol

Locations

```
688   claimWait = 3600;
689   minimumTokenBalanceForDividends = minBalance * 10 ** 9;
690   }
691
692   function _transfer(address, address, uint256) internal pure override {
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 840

low SEVERITY

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

Source File

- ELC.sol

Locations

```
839   while(gasUsed < gas && iterations < numberOfTokenHolders) {
840   _lastProcessedIndex++;
841
842   if(_lastProcessedIndex >= tokenHoldersMap.keys.length) {
843   _lastProcessedIndex = 0;
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 850

low SEVERITY

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

Source File

- ELC.sol

Locations

```
849   if(processAccount(payable(account), true)) {
850   claims++;
851   }
852   }
853
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 854

low SEVERITY

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

Source File

- ELC.sol

Locations

```
853
854   iterations++;
855
856   uint256 newGasLeft = gasleft();
857
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 948

low SEVERITY

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

Source File

- ELC.sol

Locations

```
947
948   totalFee         = marketingFee + rewardsFee + teamFee + validatorFee + buyBackFee;
949
950   marketingWallet = 0x862cD7bC48eBaB97436eE29953c62A8aaD2fBCDC;
951   teamWallet = 0xffa1cF4835b3D41A02b7e4b565186ece8eF55Be8;
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 977

low SEVERITY

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

Source File

- ELC.sol

Locations

```
976
977   _mint(owner(), 1_000_000_000 * (10 ** 9));
978   swapTokensAtAmount = totalSupply() / 5000;
979
980   priceImpactPercent = 10;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 978

low SEVERITY

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

Source File

- ELC.sol

Locations

```
977   _mint(owner(), 1_000_000_000 * (10 ** 9));
978   swapTokensAtAmount = totalSupply() / 5000;
979
980   priceImpactPercent = 10;
981   buybackThreshold = 1_000_000;
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1060

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1059
1060   totalFee = marketingFee + rewardsFee + teamFee + validatorFee + buyBackFee;
1061
1062   require(totalFee <= 10, "Buy fee cannot be more than 10%");
1063   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1113

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1112
1113   uint256 price = getPriceOfToken(1 * 10 ** decimals());
1114
1115   uint256 bnbShare = marketingFee + rewardsFee + teamFee + validatorFee +
buyBackFee;
1116
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1115

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1114
1115   uint256 bnbShare = marketingFee + rewardsFee + teamFee + validatorFee +
buyBackFee;
1116
1117   if(contractTokenBalance > 0 && bnbShare > 0) {
1118   uint256 initialBalance = address(this).balance;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1131

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1130
1131   uint256 newBalance = address(this).balance - initialBalance;
1132
1133   if(marketingFee > 0) {
1134   uint256 marketingBNB = newBalance * marketingFee / bnbShare;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1134

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1133   if(marketingFee > 0) {
1134   uint256 marketingBNB = newBalance * marketingFee / bnbShare;
1135   sendBNB(payable(marketingWallet), marketingBNB);
1136   }
1137
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1139

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1138   if(teamFee > 0) {
1139   uint256 teamBNB = newBalance * teamFee / bnbShare;
1140   sendBNB(payable(teamWallet), teamBNB);
1141   }
1142
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1144

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1143   if(validatorFee > 0) {
1144   uint256 validatorBNB = newBalance * validatorFee / bnbShare;
1145   sendBNB(payable(validatorWallet), validatorBNB);
1146   }
1147
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1149

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1148   if(buyBackFee > 0) {
1149   buybackBNB += newBalance * buyBackFee / bnbShare;
1150
1151   if (buybackBNB > buybackThreshold && buybackEnabled && (price <= allTimeHigh *
(100 - priceImpactPercent) / 100)){
1152   buyBackTokens(buybackBNB);
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1151

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1150
1151   if (buybackBNB > buybackThreshold && buybackEnabled && (price <= allTimeHigh *
(100 - priceImpactPercent) / 100)){
1152   buyBackTokens(buybackBNB);
1153   buybackBNB = 0;
1154   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1158

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1157   if(rewardsFee > 0) {
1158   uint256 rewardBNB = newBalance * rewardsFee / bnbShare;
1159   swapAndSendDividends(rewardBNB);
1160   }
1161   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1183

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1182
1183   uint256 price = getPriceOfToken(1 * 10 ** decimals());
1184
1185   if (price >= allTimeHigh) {
1186   allTimeHigh = price;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1191

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1190   _totalFees = totalFee;
1191   require (block.timestamp - traded[from] >= cooldown, "Wait before selling.
Cooldown enabled.");
1192
1193   uint256 price = getPriceOfToken(1 * 10 ** decimals());
1194
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1193

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1192
1193   uint256 price = getPriceOfToken(1 * 10 ** decimals());
1194
1195   if (buybackBNB > buybackThreshold && buybackEnabled && (price <= allTimeHigh *
(100 - priceImpactPercent) / 100)){
1196   buyBackTokens(buybackBNB);
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1195

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1194
1195   if (buybackBNB > buybackThreshold && buybackEnabled && (price <= allTimeHigh *
(100 - priceImpactPercent) / 100)){
1196   buyBackTokens(buybackBNB);
1197   buybackBNB = 0;
1198   }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1202

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1201   }
1202   uint256 fees = amount * _totalFees / 100;
1203
1204   amount = amount - fees;
1205
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1204

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1203
1204   amount = amount - fees;
1205
1206   super._transfer(from, address(this), fees);
1207   }
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1269

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1268
1269   price = (uniswapV2Router.getAmountsOut(amount, path))[path.length - 1];
1270   }
1271
1272   function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner{
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1273

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1272   function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner{
1273   require(newAmount > totalSupply() / 100_000, "SwapTokensAtAmount must be greater
than 0.001% of total supply");
1274   swapTokensAtAmount = newAmount;
1275   }
1276
```

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

LINE 199

low SEVERITY

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

Source File

- ELC.sol

Locations

```
198   uint index = map.indexOf[key];
199   uint lastIndex = map.keys.length - 1;
200   address lastKey = map.keys[lastIndex];
201
202   map.indexOf[lastKey] = index;
```

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

LINE 1269

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1268
1269   price = (uniswapV2Router.getAmountsOut(amount, path))[path.length - 1];
1270   }
1271
1272   function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner{
```

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

LINE 1218

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

- ELC.sol

Locations

```
1217   try dividendTracker.process(gas) returns (uint256 iterations, uint256 claims,
uint256 lastProcessedIndex) {
1218   emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, true, gas,
tx.origin);
1219   }
1220   catch {
1221
```

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

LINE 1345

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

- ELC.sol

Locations

```
1344   (uint256 iterations, uint256 claims, uint256 lastProcessedIndex) =
dividendTracker.process(gas);
1345   emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, false, gas,
tx.origin);
1346   }
1347
1348   function claim() external {
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 172

low SEVERITY

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

Source File

- ELC.sol

Locations

```
171   function getKeyAtIndex(Map storage map, uint index) public view returns (address) {
172   return map.keys[index];
173   }
174
175   function size(Map storage map) public view returns (uint) {
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 200

low SEVERITY

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

Source File

- ELC.sol

Locations

```
199   uint lastIndex = map.keys.length - 1;
200   address lastKey = map.keys[lastIndex];
201
202   map.indexOf[lastKey] = index;
203   delete map.indexOf[key];
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 205

low SEVERITY

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

Source File

- ELC.sol

Locations

```
204
205   map.keys[index] = lastKey;
206   map.keys.pop();
207   }
208   }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 846

low SEVERITY

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

Source File

- ELC.sol

Locations

```
845
846   address account = tokenHoldersMap.keys[_lastProcessedIndex];
847
848   if(canAutoClaim(lastClaimTimes[account])) {
849   if(processAccount(payable(account), true)) {
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1121

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1120   address[] memory path = new address[](2);
1121   path[0] = address(this);
1122   path[1] = uniswapV2Router.WETH();
1123
1124   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1122

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1121   path[0] = address(this);
1122   path[1] = uniswapV2Router.WETH();
1123
1124   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
1125   contractTokenBalance,
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1228

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1227   address[] memory path = new address[](2);
1228   path[0] = uniswapV2Router.WETH();
1229   path[1] = rewardToken;
1230
1231   uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1229

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1228   path[0] = uniswapV2Router.WETH();
1229   path[1] = rewardToken;
1230
1231   uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(
1232   0,
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1249

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1248   address[] memory path = new address[](2);
1249   path[0] = uniswapV2Router.WETH();
1250   path[1] = address(this);
1251
1252   uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
_amount}(
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1250

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1249   path[0] = uniswapV2Router.WETH();
1250   path[1] = address(this);
1251
1252   uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
_amount}(
1253   0,
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1266

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1265   address[] memory path = new address[](2);
1266   path[0] = address(this);
1267   path[1] = uniswapV2Router.WETH();
1268
1269   price = (uniswapV2Router.getAmountsOut(amount, path))[path.length - 1];
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1267

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1266   path[0] = address(this);
1267   path[1] = uniswapV2Router.WETH();
1268
1269   price = (uniswapV2Router.getAmountsOut(amount, path))[path.length - 1];
1270   }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1269

low SEVERITY

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

Source File

- ELC.sol

Locations

```
1268
1269   price = (uniswapV2Router.getAmountsOut(amount, path))[path.length - 1];
1270   }
1271
1272   function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner{
```

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.