Luzion Protocol

Smart Contract Audit Report

SYSFIXED

12 Apr 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Luzion Protocol	LZN	Binance Smart Chain

Addresses

Contract address	0x291c4e4277f8717e0552d108dbd7f795a9fef016
Contract deployer address	0x354e77bC87c8b1ff4fF00EF62f88829f23d44aD5

Project Website

https://www.luzion.app/

Codebase

https://bscscan.com/address/0x291c4e4277f8717e0552d108dbd7f795a9fef016#code

SUMMARY

The Luzion Protocol is a decentralized financial asset developed by the Revoluzion Ecosystem. The team members are fully transparent and committed to creating a trustworthy and reliable project. The Luzion Protocol utilizes the unique Auto-Staking Protocol and Auto-Reflection (ASPAR) protocol to offer a sustainable fixed compound interest model to token holders. The ASPAR protocol automatically stakes the Luzion Protocol token and offers features such as BUSD rewards and the highest Fixed APY in the market at 383,125.80% for the first 12 months. The Luzion Protocol team consists of 12 experienced and skilled developers, marketers, and operations professionals, who are dedicated to providing a fully functional protocol in the DeFi space for the community. One of the key benefits of the Luzion Protocol is its ease and safety of staking. The Auto staking feature allows users to receive rewards directly in their wallet without the need for complicated staking processes. Additionally, 4% of all trading fees are stored in the Luzion Protocol Dividend Fund (LPDF), which helps to maximize profitability, stability, and long-term sustainability. The Luzion Protocol also boasts the fastest auto-compounding rate in crypto, with payouts to token holders every 15 minutes, or 96 times per day. To ensure that the circulating supply of the token remains manageable, the Luzion Protocol features an automatic token burn system called "The Black Hole," which depletes 2% of Luzion Protocol tokens from transactions indefinitely. In addition to these features, the Luzion Protocol offers the highest Fixed APY at 383,125.80% for the first 12 months, followed by a predefined Long-term Interest Cycle period. Overall, the Luzion Protocol is a powerful and innovative DeFi asset offering exceptional returns and benefits to token holders.

Contract Summary

Documentation Quality

Luzion Protocol 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 Luzion Protocol with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 41, 55, 70, 71, 84, 96, 111, 125, 139, 153, 169, 192, 215, 241, 683, 706, 735, 737, 759, 760, 785, 787, 828, 1085, 1087, 1087, 1200, 1201, 1209, 1267, 1268, 1395, 1397, 1398, 1398, 1518, 1519, 1525, 1525, 1527, 1527, 1527, 1529, 1533, 1534, 1534, 1835, 1842, 1267 and 1268.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 16.
- 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 1057, 1162, 1163, 1194, 1195, 1267, 1267, 1268, 1600, 1601, 1642, 1643, 1671, 1672, 1901, 1901, 1901 and 1901.

CONCLUSION

We have audited the Luzion Protocol project released on April 2022 to discover issues and identify potential security vulnerabilities in Luzion Protocol Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

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

The issues found in the Luzion Protocol 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 public state variable with array type causing reachable exception by default, 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. The current pragma Solidity directive is ""^0.8.13"". Specifying a fixed compiler version is recommended to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

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.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

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

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

SMART CONTRACT ANALYSIS

Started	Monday Apr 11 2022 14:18:29 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Apr 12 2022 04:52:18 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	LuzionProtocol.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

SYSFIXED

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-103	A FLOATING PRAGMA IS SET.	low	acknowledged

SWC-110	PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 41

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
40   unchecked {
41   uint256 c = a + b;
42   if (c < a) return (false, 0);
43   return (true, c);
44   }
45
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 55

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
if (b > a) return (false, 0);
return (true, a - b);
}
}


```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 70

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
69   if (a == 0) return (true, 0);
70   uint256 c = a * b;
71   if (c / a != b) return (false, 0);
72   return (true, c);
73   }
74
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 71

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
70   uint256 c = a * b;
71   if (c / a != b) return (false, 0);
72   return (true, c);
73   }
74   }
75
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 84

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
83   if (b == 0) return (false, 0);
84   return (true, a / b);
85   }
86   }
87
88
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 96

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
95    if (b == 0) return (false, 0);
96    return (true, a % b);
97    }
98    }
99
100
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 111

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
110   function add(uint256 a, uint256 b) internal pure returns (uint256) {
111   return a + b;
112   }
113
114   /**
115
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 125

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
124   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
125   return a - b;
126   }
127
128   /**
129
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 139

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
138   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
139   return a * b;
140   }
141
142   /**
143
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 153

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
152   function div(uint256 a, uint256 b) internal pure returns (uint256) {
153   return a / b;
154   }
155
156   /**
157
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 169

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
168   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
169   return a % b;
170   }
171
172   /**
173
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 192

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
191   require(b <= a, errorMessage);
192   return a - b;
193   }
194   }
195
196
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 215

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
214   require(b > 0, errorMessage);
215   return a / b;
216   }
217   }
218
219
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 241

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
240   require(b > 0, errorMessage);
241   return a % b;
242   }
243   }
244   }
245
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 683

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
address owner = _msgSender();
_approve(owner, spender, allowance(owner, spender) + addedValue);
return true;
}


```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 706

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
705   unchecked {
706   _approve(owner, spender, currentAllowance - subtractedValue);
707   }
708
709   return true;
710
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 735

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
734   unchecked {
735   _balances[from] = fromBalance - amount;
736   }
737   _balances[to] += amount;
738
739
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 737

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
736   }
737   _balances[to] += amount;
738
739   emit Transfer(from, to, amount);
740
741
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 759

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
758
759   _totalSupply += amount;
760   _balances[account] += amount;
761   emit Transfer(address(0), account, amount);
762
763
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 760

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
759   _totalSupply += amount;
760   _balances[account] += amount;
761   emit Transfer(address(0), account, amount);
762
763   _afterTokenTransfer(address(0), account, amount);
764
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 785

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
784   unchecked {
785   _balances[account] = accountBalance - amount;
786   }
787   _totalSupply -= amount;
788
789
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 787

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
786   }
787   _totalSupply -= amount;
788
789   emit Transfer(account, address(0), amount);
790
791
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 828

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
827   unchecked {
828   _approve(owner, spender, currentAllowance - amount);
829   }
830   }
831   }
832
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1085

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1084
1085   dividendsPerShareAccuracyFactor = 10**36;
1086   minPeriod = 1 hours;
1087   minDistribution = 1 * (10**rewardToken.decimals());
1088   }
1089
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1087

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1086   minPeriod = 1 hours;
1087   minDistribution = 1 * (10**rewardToken.decimals());
1088   }
1089
1090
1091
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1087

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1086    minPeriod = 1 hours;
1087    minDistribution = 1 * (10**rewardToken.decimals());
1088    }
1089
1090
1091
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1200

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1199   gasLeft = gasleft();
1200   currentIndex++;
1201   iterations++;
1202   }
1203   }
1204
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1201

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1200   currentIndex++;
1201   iterations++;
1202   }
1203   }
1204
1205
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1209

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1208   function shouldDistribute(address shareholder) internal view returns (bool) {
1209   return shareholderClaims[shareholder] + minPeriod < block.timestamp &&
getUnpaidEarnings(shareholder) > minDistribution;
1210   }
1211
1212   /**
1213
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1267

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1266   function removeShareholder(address shareholder) internal {
1267   shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length -
1];
1268   shareholderIndexes[shareholders[shareholders.length - 1]] =
shareholderIndexes[shareholder];
1269   shareholders.pop();
1270   }
1271
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1268

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1267   shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length -
1];
1268   shareholderIndexes[shareholders[shareholders.length - 1]] =
shareholderIndexes[shareholder];
1269   shareholders.pop();
1270   }
1271
1272
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1395

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1394   totalFee =
liquidityFee.add(treasuryFee).add(ecosystemFee).add(dividendFee).add(autoBlackholeFee);
1395   supplyInitialFragment = _supplyInitial.mul(10**5);
1396   supplyTotal = supplyInitialFragment;
1397   supplyMax = _supplyMax.mul(10**5);
1398   gonsTotal = uintMax - (uintMax % supplyInitialFragment);
1399
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1397

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1396   supplyTotal = supplyInitialFragment;
1397   supplyMax = _supplyMax.mul(10**5);
1398   gonsTotal = uintMax - (uintMax % supplyInitialFragment);
1399   gonsPerFragment = gonsTotal.div(supplyTotal);
1400   gonSwapThreshold = gonsTotal.div(10000).mul(10);
1401
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1398

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1397   supplyMax = _supplyMax.mul(10**5);
1398   gonsTotal = uintMax - (uintMax % supplyInitialFragment);
1399   gonsPerFragment = gonsTotal.div(supplyTotal);
1400   gonSwapThreshold = gonsTotal.div(10000).mul(10);
1401
1402
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1398

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1397   supplyMax = _supplyMax.mul(10**5);
1398   gonsTotal = uintMax - (uintMax % supplyInitialFragment);
1399   gonsPerFragment = gonsTotal.div(supplyTotal);
1400   gonSwapThreshold = gonsTotal.div(10000).mul(10);
1401
1402
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1518

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1517
1518   uint256 deltaTimeFromInit = block.timestamp - initRebaseStartTime;
1519   uint256 deltaTime = block.timestamp - lastRebasedTime;
1520   uint256 times = deltaTime.div(15 minutes);
1521   uint256 epoch = times.mul(15);
1522
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1519

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1518   uint256 deltaTimeFromInit = block.timestamp - initRebaseStartTime;
1519   uint256 deltaTime = block.timestamp - lastRebasedTime;
1520   uint256 times = deltaTime.div(15 minutes);
1521   uint256 epoch = times.mul(15);
1522
1523
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1525

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1524   rebaseRate = 2355;
1525   } else if (deltaTimeFromInit >= (365 days) && deltaTimeFromInit < ((15 * 365 days)
/ 10)) {
1526   rebaseRate = 211;
1527   } else if (deltaTimeFromInit >= ((15 * 365 days) / 10) && deltaTimeFromInit < (7 *
365 days)) {
1528   rebaseRate = 14;
1529
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1525

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1524   rebaseRate = 2355;
1525   } else if (deltaTimeFromInit >= (365 days) && deltaTimeFromInit < ((15 * 365 days)
/ 10)) {
1526   rebaseRate = 211;
1527   } else if (deltaTimeFromInit >= ((15 * 365 days) / 10) && deltaTimeFromInit < (7 *
365 days)) {
1528   rebaseRate = 14;
1529
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1527

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1526   rebaseRate = 211;
1527   } else if (deltaTimeFromInit >= ((15 * 365 days) / 10) && deltaTimeFromInit < (7 *
365 days)) {
1528   rebaseRate = 14;
1529   } else if (deltaTimeFromInit >= (7 * 365 days)) {
1530   rebaseRate = 2;
1531
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1527

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1526   rebaseRate = 211;
1527   } else if (deltaTimeFromInit >= ((15 * 365 days) / 10) && deltaTimeFromInit < (7 *
365 days)) {
1528   rebaseRate = 14;
1529   } else if (deltaTimeFromInit >= (7 * 365 days)) {
1530   rebaseRate = 2;
1531
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1527

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1526   rebaseRate = 211;
1527   } else if (deltaTimeFromInit >= ((15 * 365 days) / 10) && deltaTimeFromInit < (7 *
365 days)) {
1528   rebaseRate = 14;
1529   } else if (deltaTimeFromInit >= (7 * 365 days)) {
1530   rebaseRate = 2;
1531
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1529

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1528   rebaseRate = 14;
1529   } else if (deltaTimeFromInit >= (7 * 365 days)) {
1530   rebaseRate = 2;
1531   }
1532
1533
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1533

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1532
1533   for (uint256 i = 0; i < times; i++) {
1534   supplyTotal =
supplyTotal.mul((10**rateDecimals).add(rebaseRate)).div(10**rateDecimals);
1535   }
1536
1537
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1534

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1533   for (uint256 i = 0; i < times; i++) {
1534   supplyTotal =
supplyTotal.mul((10**rateDecimals).add(rebaseRate)).div(10**rateDecimals);
1535   }
1536
1537   gonsPerFragment = gonsTotal.div(supplyTotal);
1538
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1534

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1533   for (uint256 i = 0; i < times; i++) {
1534   supplyTotal =
supplyTotal.mul((10**rateDecimals).add(rebaseRate)).div(10**rateDecimals);
1535   }
1536
1537   gonsPerFragment = gonsTotal.div(supplyTotal);
1538
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1835

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1834   function shouldRebase() internal view returns (bool) {
1835   return autoRebase && (supplyTotal < supplyMax) && _msgSender() != pair  && !inSwap
&& block.timestamp >= (lastRebasedTime + 15 minutes);
1836   }
1837
1838   /**
1839
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1842

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1841   function shouldAddLiquidity() internal view returns (bool) {
1842   return autoAddLiquidity && !inSwap && _msgSender() != pair && block.timestamp >=
(lastAddLiquidityTime + 12 hours);
1843   }
1844
1845   /**
1846
```

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

LINE 1267

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1266   function removeShareholder(address shareholder) internal {
1267   shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length - 1];
1268   shareholderIndexes[shareholders[shareholders.length - 1]] = shareholderIndexes[shareholder];
1269   shareholders.pop();
1270   }
1271
```

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

LINE 1268

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1267   shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length -
1];
1268   shareholderIndexes[shareholders[shareholders.length - 1]] =
shareholderIndexes[shareholder];
1269   shareholders.pop();
1270   }
1271
1272
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 16

low SEVERITY

The current pragma Solidity directive is ""^0.8.13"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- LuzionProtocol.sol

Locations

```
15
16   pragma solidity ^0.8.13;
17
18
19   /** LIBRARY / DEPENDENCY **/
20
```

SWC-110 | PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.

LINE 1057

low SEVERITY

The public state variable "shareholders" in "DividendDistributor" contract has type "address[]" and can cause an exception in case of use of invalid array index value.

Source File

- LuzionProtocol.sol

Locations

```
1056   address public _token;
1057   address[] public shareholders;
1058
1059   mapping(address => uint256) public shareholderIndexes;
1060   mapping(address => uint256) public shareholderClaims;
1061
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1162

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1161   address[] memory path = new address[](2);
1162   path[0] = router.WETH();
1163   path[1] = address(rewardToken);
1164
1165   router.swapExactETHForTokensSupportingFeeOnTransferTokens {
1166
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1163

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1162   path[0] = router.WETH();
1163   path[1] = address(rewardToken);
1164
1165   router.swapExactETHForTokensSupportingFeeOnTransferTokens {
1166   value: _msgValue()
1167
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1194

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1193
1194   if (shouldDistribute(shareholders[currentIndex])) {
1195   distributeDividend(shareholders[currentIndex]);
1196   }
1197
1198
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1195

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1194   if (shouldDistribute(shareholders[currentIndex])) {
1195   distributeDividend(shareholders[currentIndex]);
1196   }
1197
1198   gasUsed = gasUsed.add(gasLeft.sub(gasleft()));
1199
```

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

- LuzionProtocol.sol

Locations

```
1266   function removeShareholder(address shareholder) internal {
1267   shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length - 1];
1268   shareholderIndexes[shareholders[shareholders.length - 1]] = shareholderIndexes[shareholder];
1269   shareholders.pop();
1270   }
1271
```

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

- LuzionProtocol.sol

Locations

```
1266   function removeShareholder(address shareholder) internal {
1267   shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length -
1];
1268   shareholderIndexes[shareholders[shareholders.length - 1]] =
shareholderIndexes[shareholder];
1269   shareholders.pop();
1270   }
1271
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1268

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1267   shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length -
1];
1268   shareholderIndexes[shareholders[shareholders.length - 1]] =
shareholderIndexes[shareholder];
1269   shareholders.pop();
1270   }
1271
1272
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1600

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1599   address[] memory path = new address[](2);
1600   path[0] = address(this);
1601   path[1] = router.WETH();
1602
1603   router.swapExactTokensForETHSupportingFeeOnTransferTokens(amountToSwap, 0, path,
address(this), block.timestamp);
1604
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1601

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1600   path[0] = address(this);
1601   path[1] = router.WETH();
1602
1603   router.swapExactTokensForETHSupportingFeeOnTransferTokens(amountToSwap, 0, path,
address(this), block.timestamp);
1604
1605
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1642

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1641   address[] memory path = new address[](2);
1642   path[0] = address(this);
1643   path[1] = router.WETH();
1644
1645   uint256 balanceBefore = address(this).balance;
1646
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1643

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1642    path[0] = address(this);
1643    path[1] = router.WETH();
1644
1645    uint256 balanceBefore = address(this).balance;
1646
1647
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1671

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1670   address[] memory path = new address[](2);
1671   path[0] = router.WETH();
1672   path[1] = address(this);
1673
1674   router.swapExactETHForTokensSupportingFeeOnTransferTokens {
1675
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1672

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1671   path[0] = router.WETH();
1672   path[1] = address(this);
1673
1674   router.swapExactETHForTokensSupportingFeeOnTransferTokens {
1675   value: amount
1676
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1901

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1900   function _initializeFeeReceivers(address[4] memory _feeReceiverSettings) internal
{
1901   _setFeeReceivers(_feeReceiverSettings[0], _feeReceiverSettings[1],
_feeReceiverSettings[2], _feeReceiverSettings[3]);
1902   }
1903
1904   /**
1905
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1901

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1900   function _initializeFeeReceivers(address[4] memory _feeReceiverSettings) internal
{
1901   _setFeeReceivers(_feeReceiverSettings[0], _feeReceiverSettings[1],
_feeReceiverSettings[2], _feeReceiverSettings[3]);
1902   }
1903
1904   /**
1905
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1901

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1900   function _initializeFeeReceivers(address[4] memory _feeReceiverSettings) internal
{
1901   _setFeeReceivers(_feeReceiverSettings[0], _feeReceiverSettings[1],
_feeReceiverSettings[2], _feeReceiverSettings[3]);
1902   }
1903
1904   /**
1905
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1901

low SEVERITY

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

Source File

- LuzionProtocol.sol

Locations

```
1900   function _initializeFeeReceivers(address[4] memory _feeReceiverSettings) internal
{
1901   _setFeeReceivers(_feeReceiverSettings[0], _feeReceiverSettings[1],
_feeReceiverSettings[2], _feeReceiverSettings[3]);
1902   }
1903
1904   /**
1905
```

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.