LuckyBlock

Smart Contract Audit Report

SYSFIXED

04 Dec 2021

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
LuckyBlock	LBlock	Binance Smart Chain

Addresses

Contract address	0x2cd96e8c3ff6b5e01169f6e3b61d28204e7810bb
Contract deployer address	0x01103B62a82071442Aa56F1Fb496b9C0c8844797

Project Website

https://www.luckyblock.com/id

Codebase

https://bscscan.com/address/0x2cd96e8c3ff6b5e01169f6e3b61d28204e7810bb#code

SUMMARY

Lucky Block casino and sportsbook is reinventing the way people bet - with the $LBlock token being the fastest growing cryptocurrency to reach a $1 billion market cap. Our casino and sportsbook, provides the best of both worlds - an engaging and entertaining casino experience, combined with a sportsbook that provides fantastic odds in a wide array of sporting events.

Contract Summary

Documentation Quality

LuckyBlock 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 LuckyBlock 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 767, 768 and 790.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 197, 211, 226, 227, 240, 252, 267, 281, 295, 309, 325, 348, 371, 397, 715, 720, 725, 730, 735, 745, 765, 765, 1001, 1003, 1048, 1056, 1074, 1081, 1138, 1138, 1147, 1147, 1164, 1164, 1180, 1180, 1196, 1196, 1208 and 1003.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 8.
- 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 1002, 1003, 1003, 1210, 1211, 1214, 1216, 1252 and 1253.

CONCLUSION

We have audited the LuckyBlock project released on December 2021 to discover issues and identify potential security vulnerabilities in LuckyBlock 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 LuckyBlock smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, a state variable visibility is not set, and out-of-bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value. The current pragma Solidity directive is ""^0.8.0"". 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. It is best practice to set the visibility of state variables explicitly. The default visibility for "isTaxless" is internal. Other possible visibility settings are public and private.

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.	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	Friday Dec 03 2021 22:32:11 GMT+0000 (Coordinated Universal Time)
Finished	Saturday Dec 04 2021 14:53:34 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	LuckyBlock.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 197

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
196   unchecked {
197   uint256 c = a + b;
198   if (c < a) return (false, 0);
199   return (true, c);
200   }
201
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 211

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
210   if (b > a) return (false, 0);
211   return (true, a - b);
212   }
213   }
214
215
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 226

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
225   if (a == 0) return (true, 0);
226   uint256 c = a * b;
227   if (c / a != b) return (false, 0);
228   return (true, c);
229   }
230
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 227

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
226   uint256 c = a * b;
227   if (c / a != b) return (false, 0);
228   return (true, c);
229   }
230   }
231
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 240

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
239   if (b == 0) return (false, 0);
240   return (true, a / b);
241   }
242   }
243
244
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 252

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

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

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 267

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
266   function add(uint256 a, uint256 b) internal pure returns (uint256) {
267   return a + b;
268   }
269
270   /**
271
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 281

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
280   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
281   return a - b;
282   }
283
284   /**
285
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 295

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
294   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
295   return a * b;
296   }
297
298   /**
299
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 309

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
308   function div(uint256 a, uint256 b) internal pure returns (uint256) {
309   return a / b;
310   }
311
312   /**
313
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 325

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

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

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 348

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
347   require(b <= a, errorMessage);
348   return a - b;
349   }
350   }
351
352
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 371

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
370   require(b > 0, errorMessage);
371   return a / b;
372   }
373   }
374
375
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 397

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
396   require(b > 0, errorMessage);
397   return a % b;
398   }
399   }
400   }
401
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 715

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
714   function toUint128(uint256 value) internal pure returns (uint128) {
715   require(value < 2**128, "SafeCast: value doesn't fit in 128 bits");
716   return uint128(value);
717   }
718
719
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 720

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
719   function toUint64(uint256 value) internal pure returns (uint64) {
720   require(value < 2**64, "SafeCast: value doesn't fit in 64 bits");
721   return uint64(value);
722   }
723
724
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 725

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
724   function toUint32(uint256 value) internal pure returns (uint32) {
725   require(value < 2**32, "SafeCast: value doesn't fit in 32 bits");
726   return uint32(value);
727   }
728
729
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 730

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
729   function toUint16(uint256 value) internal pure returns (uint16) {
730   require(value < 2**16, "SafeCast: value doesn't fit in 16 bits");
731   return uint16(value);
732   }
733
734
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 735

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
734   function toUint8(uint256 value) internal pure returns (uint8) {
735   require(value < 2**8, "SafeCast: value doesn't fit in 8 bits");
736   return uint8(value);
737   }
738
739
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 745

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
744   function toInt256(uint256 value) internal pure returns (int256) {
745   require(value < 2**255, "SafeCast: value doesn't fit in an int256");
746   return int256(value);
747   }
748   }
749
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 765

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
764   uint256 internal _tokenTotal = 100_000_000_000e9;
765   uint256 internal _reflectionTotal = (MAX - (MAX % _tokenTotal));
766
767   mapping(address => bool) blacklist;
768   mapping(address => bool) isTaxless;
769
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 765

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
764   uint256 internal _tokenTotal = 100_000_000_000e9;
765   uint256 internal _reflectionTotal = (MAX - (MAX % _tokenTotal));
766
767   mapping(address => bool) blacklist;
768   mapping(address => bool) isTaxless;
769
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1001

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1000   require(_isExcluded[account], "TOKEN: Account is already included");
1001   for (uint256 i = 0; i < _excluded.length; i++) {
1002   if (_excluded[i] == account) {
1003   _excluded[i] = _excluded[_excluded.length - 1];
1004   _tokenBalance[account] = 0;
1005
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1003

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1002   if (_excluded[i] == account) {
1003   _excluded[i] = _excluded[_excluded.length - 1];
1004   _tokenBalance[account] = 0;
1005   _isExcluded[account] = false;
1006   _excluded.pop();
1007
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1048

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1047   !isTaxless[recipient] &&
1048   listedAt + 3 minutes >= block.timestamp
1049   ) {
1050   // don't allow to buy more than 0.01% of total supply for 3 minutes after launch
1051   require(
1052
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1056

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1055   );
1056   if (listedAt + 180 seconds >= block.timestamp)
1057   // don't allow sell for 180 seconds afer launch
1058   require(
1059   recipient != pancakeSwapV2Pair,
1060
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1074

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1073   if (
1074   lastSellCycleStart[sender] + sellCooldownTime < block.timestamp
1075   ) {
1076   lastSellCycleStart[sender] = block.timestamp;
1077   transferAmounts[sender] = 0;
1078
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1081

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1080   if (transferAmounts[sender] >= sellCooldownAmount) {
1081   sellCooldown[sender] = block.timestamp + sellCooldownTime;
1082   }
1083   }
1084
1085
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1138

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1137   if(taxFee != 0 && to == distributionWallet){
1138   uint256 _taxFee = amount.mul(taxFee).div(10**(feeDecimal + 2));
1139   transferAmount = transferAmount.sub(_taxFee);
1140   _reflectionTotal = _reflectionTotal.sub(_taxFee.mul(rate));
1141   taxFeeTotal = taxFeeTotal.add(_taxFee);
1142
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1138

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1137   if(taxFee != 0 && to == distributionWallet){
1138   uint256 _taxFee = amount.mul(taxFee).div(10**(feeDecimal + 2));
1139   transferAmount = transferAmount.sub(_taxFee);
1140   _reflectionTotal = _reflectionTotal.sub(_taxFee.mul(rate));
1141   taxFeeTotal = taxFeeTotal.add(_taxFee);
1142
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1147

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1146   uint256 _liquidityFee = amount.mul(liquidityFee).div(
1147   10**(feeDecimal + 2)
1148   );
1149   transferAmount = transferAmount.sub(_liquidityFee);
1150   _reflectionBalance[address(this)] = _reflectionBalance[
1151
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1147

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1146   uint256 _liquidityFee = amount.mul(liquidityFee).div(
1147   10**(feeDecimal + 2)
1148   );
1149   transferAmount = transferAmount.sub(_liquidityFee);
1150   _reflectionBalance[address(this)] = _reflectionBalance[
1151
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1164

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1163   if (lotteryPoolFee != 0 && to == pancakeSwapV2Pair) {
1164   uint256 _lotteryPoolFee = amount.mul(lotteryPoolFee).div(10**(feeDecimal + 2));
1165   transferAmount = transferAmount.sub(_lotteryPoolFee);
1166   _reflectionBalance[lotteryPoolWallet] = _reflectionBalance[lotteryPoolWallet].add(
1167   _lotteryPoolFee.mul(rate)
1168
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1164

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1163   if (lotteryPoolFee != 0 && to == pancakeSwapV2Pair) {
1164   uint256 _lotteryPoolFee = amount.mul(lotteryPoolFee).div(10**(feeDecimal + 2));
1165   transferAmount = transferAmount.sub(_lotteryPoolFee);
1166   _reflectionBalance[lotteryPoolWallet] = _reflectionBalance[lotteryPoolWallet].add(
1167   _lotteryPoolFee.mul(rate)
1168
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1180

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1179   if (nftRoyalPoolFee != 0 && to == pancakeSwapV2Pair) {
1180   uint256 _nftRoyalPoolFee = amount.mul(nftRoyalPoolFee).div(10**(feeDecimal + 2));
1181   transferAmount = transferAmount.sub(_nftRoyalPoolFee);
1182   _reflectionBalance[nftRoyalPoolWallet] =
_reflectionBalance[nftRoyalPoolWallet].add(
1183   _nftRoyalPoolFee.mul(rate)
1184
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1180

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1179   if (nftRoyalPoolFee != 0 && to == pancakeSwapV2Pair) {
1180   uint256 _nftRoyalPoolFee = amount.mul(nftRoyalPoolFee).div(10**(feeDecimal + 2));
1181   transferAmount = transferAmount.sub(_nftRoyalPoolFee);
1182   _reflectionBalance[nftRoyalPoolWallet] =
_reflectionBalance[nftRoyalPoolWallet].add(
1183   _nftRoyalPoolFee.mul(rate)
1184
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1196

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1195   if (burnFee != 0 && to == pancakeSwapV2Pair) {
1196   uint256 _burnFee = amount.mul(burnFee).div(10**(feeDecimal + 2));
1197   transferAmount = transferAmount.sub(_burnFee);
1198   _tokenBalance[deadWallet] = _tokenBalance[deadWallet].add(_burnFee);
1199   emit Transfer(account, deadWallet, _burnFee);
1200
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1196

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1195   if (burnFee != 0 && to == pancakeSwapV2Pair) {
1196   uint256 _burnFee = amount.mul(burnFee).div(10**(feeDecimal + 2));
1197   transferAmount = transferAmount.sub(_burnFee);
1198   _tokenBalance[deadWallet] = _tokenBalance[deadWallet].add(_burnFee);
1199   emit Transfer(account, deadWallet, _burnFee);
1200
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1208

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1207   uint256 tokenSupply = _tokenTotal;
1208   for (uint256 i = 0; i < _excluded.length; i++) {
1209   if (
1210   _reflectionBalance[_excluded[i]] > reflectionSupply ||
1211   _tokenBalance[_excluded[i]] > tokenSupply
1212
```

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

LINE 1003

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1002   if (_excluded[i] == account) {
1003   _excluded[i] = _excluded[_excluded.length - 1];
1004   _tokenBalance[account] = 0;
1005   _isExcluded[account] = false;
1006   _excluded.pop();
1007
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 8

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". 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

- LuckyBlock.sol

Locations

```
7
8   pragma solidity ^0.8.0;
9
10   /**
11   * @dev Interface of the ERC20 standard as defined in the EIP.
12
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 767

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "blacklist" is internal. Other possible visibility settings are public and private.

Source File

- LuckyBlock.sol

Locations

```
766
767   mapping(address => bool) blacklist;
768   mapping(address => bool) isTaxless;
769   mapping(address => bool) internal _isExcluded;
770   address[] internal _excluded;
771
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 768

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "isTaxless" is internal. Other possible visibility settings are public and private.

Source File

- LuckyBlock.sol

Locations

```
767   mapping(address => bool) blacklist;
768   mapping(address => bool) isTaxless;
769   mapping(address => bool) internal _isExcluded;
770   address[] internal _excluded;
771
772
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 790

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "deadWallet" is internal. Other possible visibility settings are public and private.

Source File

- LuckyBlock.sol

Locations

```
789   address public lotteryPoolWallet=0x1C68DFB90D6F2acb05b7b34dF191e9C6B38E9Cb1;
790   address deadWallet = 0x000000000000000000000000000000000000dEaD;
791
792   address public admin;
793
794
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1002

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1001   for (uint256 i = 0; i < _excluded.length; i++) {
1002   if (_excluded[i] == account) {
1003   _excluded[i] = _excluded[_excluded.length - 1];
1004   _tokenBalance[account] = 0;
1005   _isExcluded[account] = false;
1006
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1003

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1002   if (_excluded[i] == account) {
1003   _excluded[i] = _excluded[_excluded.length - 1];
1004   _tokenBalance[account] = 0;
1005   _isExcluded[account] = false;
1006   _excluded.pop();
1007
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1003

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1002   if (_excluded[i] == account) {
1003   _excluded[i] = _excluded[_excluded.length - 1];
1004   _tokenBalance[account] = 0;
1005   _isExcluded[account] = false;
1006   _excluded.pop();
1007
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1210

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1209   if (
1210   _reflectionBalance[_excluded[i]] > reflectionSupply ||
1211   _tokenBalance[_excluded[i]] > tokenSupply
1212   ) return _reflectionTotal.div(_tokenTotal);
1213   reflectionSupply = reflectionSupply.sub(
1214
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1211

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1210   _reflectionBalance[_excluded[i]] > reflectionSupply ||
1211   _tokenBalance[_excluded[i]] > tokenSupply
1212   ) return _reflectionTotal.div(_tokenTotal);
1213   reflectionSupply = reflectionSupply.sub(
1214   _reflectionBalance[_excluded[i]]
1215
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1214

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1213   reflectionSupply = reflectionSupply.sub(
1214   _reflectionBalance[_excluded[i]]
1215   );
1216   tokenSupply = tokenSupply.sub(_tokenBalance[_excluded[i]]);
1217   }
1218
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1216

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1215   );
1216   tokenSupply = tokenSupply.sub(_tokenBalance[_excluded[i]]);
1217   }
1218   if (reflectionSupply < _reflectionTotal.div(_tokenTotal))
1219   return _reflectionTotal.div(_tokenTotal);
1220
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1252

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1251   address[] memory path = new address[](2);
1252   path[0] = address(this);
1253   path[1] = pancakeswapV2Router.WETH();
1254
1255   _approve(address(this), address(pancakeswapV2Router), tokenAmount);
1256
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1253

low SEVERITY

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

Source File

- LuckyBlock.sol

Locations

```
1252   path[0] = address(this);
1253   path[1] = pancakeswapV2Router.WETH();
1254
1255   _approve(address(this), address(pancakeswapV2Router), tokenAmount);
1256
1257
```

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.