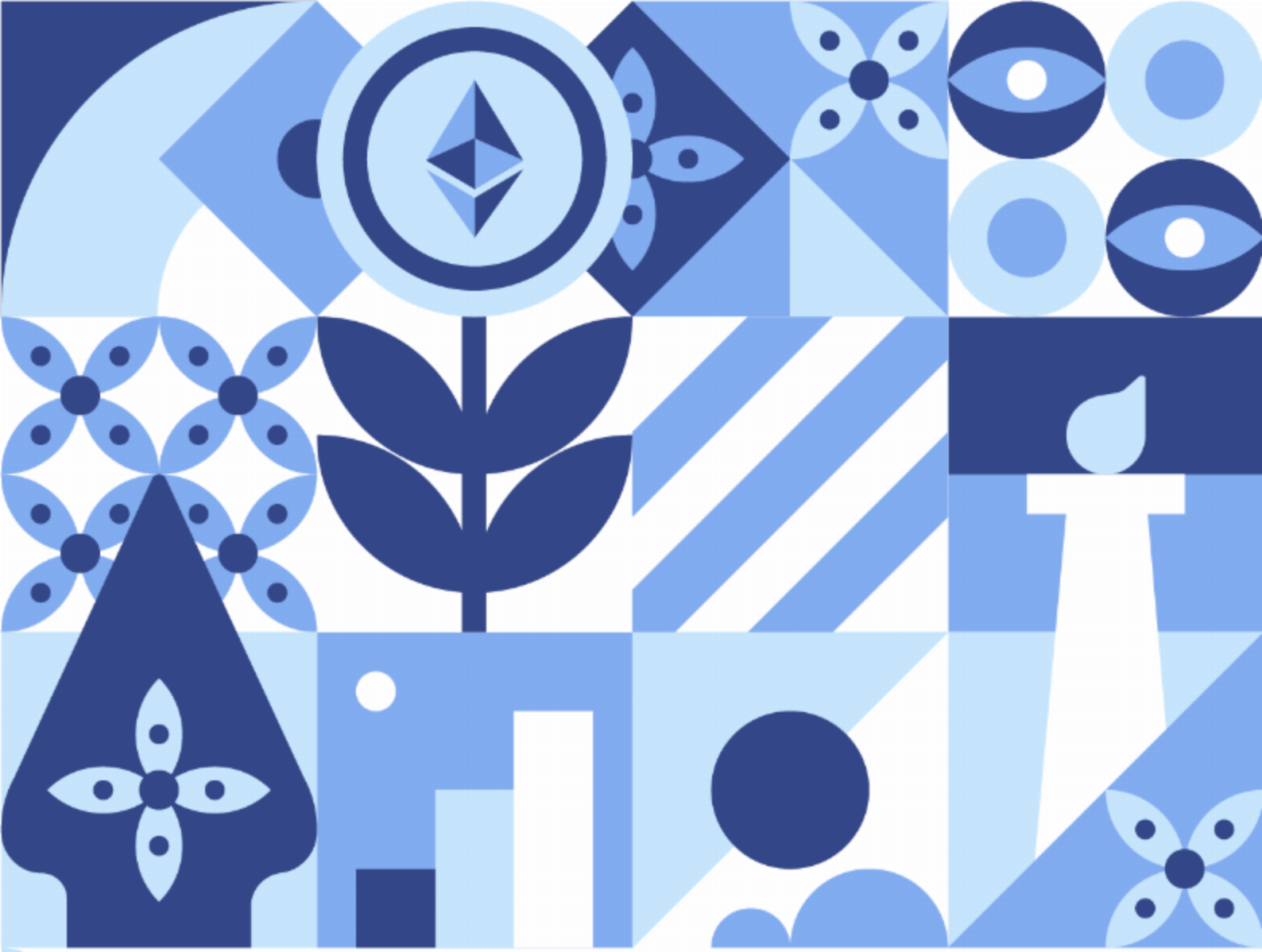

Lobby Token

Smart Contract Audit Report

SYSFIXED

22 Nov 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Lobby Token	LBY	Ethereum

Addresses

Contract address	0xac042d9284df95cc6bd35982f6a61e3e7a6f875b
Contract deployer address	0x98600d7F402950f830D510CCc9d3ead4f6109033

Project Website

https://www.lobbytoken.io/

Codebase

https://etherscan.io/address/0xac042d9284df95cc6bd35982f6a61e3e7a6f875b#code

SUMMARY

$LBY is a governance token that powers and secures the Lobby DAO. Holders of $LBY can vote on proposals for Lobby DAO as well as all future products within the Lobby DAO ecosystem.

Contract Summary

Documentation Quality

Lobby Token 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 Lobby Token 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 723 and 749.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 156, 188, 206, 206, 242, 282, 503, 753, 753, 753, 753, 753, 754, 773, 773, 773, 773, 774, 774, 774, 774, 871, 871, 883, 884, 884, 922, 924, 972, 972, 980, 980, 1052, 1083, 1085, 1188 and 924.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 51.
- 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 882, 883, 922, 923, 924, 1052, 1053, 1058, 1059, 1198 and 1199.

CONCLUSION

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

The security audit report provides a satisfactory result with some low-risk issues.

The issues in the Lobby Token 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 using an invalid array index value.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	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 Nov 21 2022 00:28:45 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Nov 22 2022 21:54:14 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Lobby.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	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-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 156

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
155   *
156   * - Subtraction cannot overflow.
157   */
158   function sub(uint256 a, uint256 b, string memory errorMessage) internal pure
returns (uint256) {
159   require(b <= a, errorMessage);
160
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 188

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
187   }
188
189   /**
190   * @dev Returns the integer division of two unsigned integers. Reverts on
191   * division by zero. The result is rounded towards zero.
192
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 206

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
205   /**
206   * @dev Returns the integer division of two unsigned integers. Reverts with custom
message on
207   * division by zero. The result is rounded towards zero.
208   *
209   * Counterpart to Solidity's `/` operator. Note: this function uses a
210
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 206

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
205   /**
206   * @dev Returns the integer division of two unsigned integers. Reverts with custom
message on
207   * division by zero. The result is rounded towards zero.
208   *
209   * Counterpart to Solidity's `/` operator. Note: this function uses a
210
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 242

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
241   /**
242   * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer
modulo),
243   * Reverts with custom message when dividing by zero.
244   *
245   * Counterpart to Solidity's `%` operator. This function uses a `revert`
246
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 282

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
281   * It is unsafe to assume that an address for which this function returns
282   * false is an externally-owned account (EOA) and not a contract.
283   *
284   * Among others, `isContract` will return false for the following
285   * types of addresses:
286
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 503

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
502
503   function getPair(address tokenA, address tokenB) external view returns (address
pair);
504   function allPairs(uint) external view returns (address pair);
505   function allPairsLength() external view returns (uint);
506
507
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 753

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
752   uint256 public _maxTxAmount = 1000000 * 10**3 * 10**9;
753   uint256 public numTokensSellToAddToLiquidity = 1000000 * 10**3 * 10**9;
754
755   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
756   event SwapAndLiquifyEnabledUpdated(bool enabled);
757
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 753

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
752   uint256 public _maxTxAmount = 1000000 * 10**3 * 10**9;
753   uint256 public numTokensSellToAddToLiquidity = 1000000 * 10**3 * 10**9;
754
755   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
756   event SwapAndLiquifyEnabledUpdated(bool enabled);
757
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 753

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
752   uint256 public _maxTxAmount = 1000000 * 10**3 * 10**9;
753   uint256 public numTokensSellToAddToLiquidity = 1000000 * 10**3 * 10**9;
754
755   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
756   event SwapAndLiquifyEnabledUpdated(bool enabled);
757
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 753

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
752   uint256 public _maxTxAmount = 1000000 * 10**3 * 10**9;
753   uint256 public numTokensSellToAddToLiquidity = 1000000 * 10**3 * 10**9;
754
755   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
756   event SwapAndLiquifyEnabledUpdated(bool enabled);
757
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 753

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
752   uint256 public _maxTxAmount = 1000000 * 10**3 * 10**9;
753   uint256 public numTokensSellToAddToLiquidity = 1000000 * 10**3 * 10**9;
754
755   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
756   event SwapAndLiquifyEnabledUpdated(bool enabled);
757
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 754

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
753   uint256 public numTokensSellToAddToLiquidity = 1000000 * 10**3 * 10**9;
754
755   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
756   event SwapAndLiquifyEnabledUpdated(bool enabled);
757   event SwapAndLiquify(
758
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 773

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
772   //IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E); //Mainnet BSC
773   //IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x9Ac64Cc6e4415144C455BD8E4837Fea55603e5c3); //Testnet BSC
774   IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D); //Mainnet & Testnet ETH
775   // Create a uniswap pair for this new token
776   uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
777
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 773

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
772   //IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E); //Mainnet BSC
773   //IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x9Ac64Cc6e4415144C455BD8E4837Fea55603e5c3); //Testnet BSC
774   IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D); //Mainnet & Testnet ETH
775   // Create a uniswap pair for this new token
776   uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
777
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 773

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
772   //IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E); //Mainnet BSC
773   //IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x9Ac64Cc6e4415144C455BD8E4837Fea55603e5c3); //Testnet BSC
774   IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D); //Mainnet & Testnet ETH
775   // Create a uniswap pair for this new token
776   uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
777
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 773

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
772   //IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E); //Mainnet BSC
773   //IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x9Ac64Cc6e4415144C455BD8E4837Fea55603e5c3); //Testnet BSC
774   IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D); //Mainnet & Testnet ETH
775   // Create a uniswap pair for this new token
776   uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
777
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 774

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
773   //IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x9Ac64Cc6e4415144C455BD8E4837Fea55603e5c3); //Testnet BSC
774   IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D); //Mainnet & Testnet ETH
775   // Create a uniswap pair for this new token
776   uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
777   .createPair(address(this), _uniswapV2Router.WETH());
778
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 774

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
773   //IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x9Ac64Cc6e4415144C455BD8E4837Fea55603e5c3); //Testnet BSC
774   IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D); //Mainnet & Testnet ETH
775   // Create a uniswap pair for this new token
776   uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
777   .createPair(address(this), _uniswapV2Router.WETH());
778
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 774

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
773   //IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x9Ac64Cc6e4415144C455BD8E4837Fea55603e5c3); //Testnet BSC
774   IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D); //Mainnet & Testnet ETH
775   // Create a uniswap pair for this new token
776   uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
777   .createPair(address(this), _uniswapV2Router.WETH());
778
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 774

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
773   //IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x9Ac64Cc6e4415144C455BD8E4837Fea55603e5c3); //Testnet BSC
774   IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D); //Mainnet & Testnet ETH
775   // Create a uniswap pair for this new token
776   uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
777   .createPair(address(this), _uniswapV2Router.WETH());
778
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 871

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
870   address sender = _msgSender();
871   require(!_isExcluded[sender], "Excluded addresses cannot call this function");
872   (uint256 rAmount,,,,,) = _getValues(tAmount);
873   _rOwned[sender] = _rOwned[sender].sub(rAmount);
874   _rTotal = _rTotal.sub(rAmount);
875
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 871

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
870   address sender = _msgSender();
871   require(!_isExcluded[sender], "Excluded addresses cannot call this function");
872   (uint256 rAmount,,,,,) = _getValues(tAmount);
873   _rOwned[sender] = _rOwned[sender].sub(rAmount);
874   _rTotal = _rTotal.sub(rAmount);
875
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 883

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
882   return rAmount;
883   } else {
884   (,uint256 rTransferAmount,,,,) = _getValues(tAmount);
885   return rTransferAmount;
886   }
887
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 884

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
883   } else {
884   (,uint256 rTransferAmount,,,,) = _getValues(tAmount);
885   return rTransferAmount;
886   }
887   }
888
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 884

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
883   } else {
884   (,uint256 rTransferAmount,,,,) = _getValues(tAmount);
885   return rTransferAmount;
886   }
887   }
888
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 922

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
921   _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount);
922   _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
923   _takeLiquidity(tLiquidity);
924   _reflectFee(rFee, tFee);
925   emit Transfer(sender, recipient, tTransferAmount);
926
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 924

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
923   _takeLiquidity(tLiquidity);
924   _reflectFee(rFee, tFee);
925   emit Transfer(sender, recipient, tTransferAmount);
926   }
927
928
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 972

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
971
972   function clearStuckBalance (address payable walletaddress) external onlyOwner() {
973   walletaddress.transfer(address(this).balance);
974   }
975
976
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 972

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
971
972   function clearStuckBalance (address payable walletaddress) external onlyOwner() {
973   walletaddress.transfer(address(this).balance);
974   }
975
976
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 980

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
979
980   function removeBotWallet(address botwallet) external onlyOwner() {
981   botWallets[botwallet] = false;
982   }
983
984
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 980

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
979
980   function removeBotWallet(address botwallet) external onlyOwner() {
981   botWallets[botwallet] = false;
982   }
983
984
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1052

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
1051
1052   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1053   return _amount.mul(_taxFee).div(
1054   10**2
1055   );
1056
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1083

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
1082
1083   function _approve(address owner, address spender, uint256 amount) private {
1084   require(owner != address(0), "ERC20: approve from the zero address");
1085   require(spender != address(0), "ERC20: approve to the zero address");
1086
1087
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1085

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
1084   require(owner != address(0), "ERC20: approve from the zero address");
1085   require(spender != address(0), "ERC20: approve to the zero address");
1086
1087   _allowances[owner][spender] = amount;
1088   emit Approval(owner, spender, amount);
1089
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 1188

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
1187   block.timestamp
1188   );
1189   }
1190
1191   function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
1192
```

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

LINE 924

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
923   _takeLiquidity(tLiquidity);
924   _reflectFee(rFee, tFee);
925   emit Transfer(sender, recipient, tTransferAmount);
926   }
927
928
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 51

low SEVERITY

The current pragma Solidity directive is ""^0.8.9"". 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

- Lobby.sol

Locations

```
50   * @dev Returns the remaining number of tokens that `spender` will be
51   * allowed to spend on behalf of `owner` through {transferFrom}. This is
52   * zero by default.
53   *
54   * This value changes when {approve} or {transferFrom} are called.
55
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 723

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
722   mapping (address => bool) private botWallets;
723   bool botscantrade = false;
724
725   bool public canTrade = false;
726   uint256 public launchTime;
727
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 749

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
748
749   bool inSwapAndLiquify;
750   bool public swapAndLiquifyEnabled = true;
751
752   uint256 public _maxTxAmount = 1000000 * 10**3 * 10**9;
753
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 882

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
881   (uint256 rAmount,,,,,) = _getValues(tAmount);
882   return rAmount;
883   } else {
884   (,uint256 rTransferAmount,,,,) = _getValues(tAmount);
885   return rTransferAmount;
886
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 883

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
882   return rAmount;
883   } else {
884   (,uint256 rTransferAmount,,,,) = _getValues(tAmount);
885   return rTransferAmount;
886   }
887
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 922

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
921   _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount);
922   _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
923   _takeLiquidity(tLiquidity);
924   _reflectFee(rFee, tFee);
925   emit Transfer(sender, recipient, tTransferAmount);
926
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 923

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
922   _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
923   _takeLiquidity(tLiquidity);
924   _reflectFee(rFee, tFee);
925   emit Transfer(sender, recipient, tTransferAmount);
926   }
927
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 924

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
923   _takeLiquidity(tLiquidity);
924   _reflectFee(rFee, tFee);
925   emit Transfer(sender, recipient, tTransferAmount);
926   }
927
928
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1052

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
1051
1052   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1053   return _amount.mul(_taxFee).div(
1054   10**2
1055   );
1056
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1053

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
1052   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1053   return _amount.mul(_taxFee).div(
1054   10**2
1055   );
1056   }
1057
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1058

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
1057
1058   function calculateLiquidityFee(uint256 _amount) private view returns (uint256) {
1059   return _amount.mul(_liquidityFee).div(
1060   10**2
1061   );
1062
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1059

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
1058   function calculateLiquidityFee(uint256 _amount) private view returns (uint256) {
1059   return _amount.mul(_liquidityFee).div(
1060   10**2
1061   );
1062   }
1063
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1198

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
1197   address(this),
1198   tokenAmount,
1199   0, // slippage is unavoidable
1200   0, // slippage is unavoidable
1201   owner(),
1202
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1199

low SEVERITY

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

Source File

- Lobby.sol

Locations

```
1198   tokenAmount,
1199   0, // slippage is unavoidable
1200   0, // slippage is unavoidable
1201   owner(),
1202   block.timestamp
1203
```

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.