Cyberscope

Audit Report

Liner Power Point

October 2023

Network	BSC
Address	0xF14fE8ea9D35cCEA2545985d7541CcAd02cb6112
Audited by	© cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Unresolved
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Unresolved

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	MSC	Missing Sanity Check	Unresolved
	MEE	Missing Events Emission	Unresolved
	RSW	Redundant Storage Writes	Unresolved
	RIS	Redundant If-Else Statement	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L20	Succeeded Transfer Check	Unresolved

Table of Contents

Review

Contract Name	LinerPowerPoint
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://bscscan.com/address/0xf14fe8ea9d35ccea2545985d7541ccad02cb6112
Address	0xf14fe8ea9d35ccea2545985d7541ccad02cb6112
Network	BSC
Symbol	LPP
Decimals	18
Total Supply	2,098,916,624

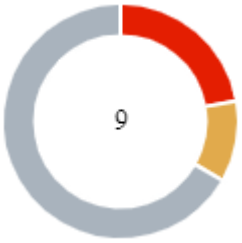
Audit Updates

Initial Audit	24 Oct 2023

Source Files

Filename	SHA256
LinerPowerPoint.sol	af08a5ab7457d445d7e8e7ee08c41cbd8e844d8fadb55349576465922e2de247

Findings Breakdown



Critical	2
Medium	1
Minor / Informative	6

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	2	0	0	0
Medium	1	0	0	0
Minor / Informative	6	0	0	0

ST - Stops Transactions

Criticality	Medium
Location	LinerPowerPoint.sol#L715,728
Status	Unresolved

Description

The contract authorized users have the authority to stop the transactions for all users excluding the authorized users. The authorized users may take advantage of it by setting the `MaxTx` to zero or `MinTx` to the max `uint256` value.

```
require(amount>= MinTx, "Error: Amount not in Min Transaction Limit");
require(amount<= MaxTx, "Error: Amount not in Max Transaction Limit");
```

The contract's owner has the authority to stop transactions for all users excluding the authorized users. The owner may take advantage of it by calling the `setPause` and the `setTrade` methods.

```
if(sender == uniswapV2Pair || recipient== uniswapV2Pair){ // Buy Sell
    if(_isExcludedFromFee[sender] || _isExcludedFromFee[recipient]){

    }else{
        require(trade, "Error: Unable to initiate your trade");
    }

}

if(_isExcludedFromFee[sender] || _isExcludedFromFee[recipient]){

}else{
    require(!pause,"Error: Unable to initiate your transfer");
    require(amount>= MinTx, "Error: Amount not in Min Transaction
Limit");
    require(amount<= MaxTx, "Error: Amount not in Max Transaction
Limit");
}
```

Recommendation

The contract could embody a check for not allowing setting the `MaxTx` less than a reasonable amount and the `MinTx` more than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply. The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	LinerPowerPoint.sol#L476
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `changeBurnFee` function with a high percentage value.

```
function changeBurnFee(uint256 _newburnFee, uint256
_newburnDivisor) external onlyOwner{
  _burnFee = _newburnFee;
  _burnDivisor = _newburnDivisor;
}
```

Recommendation

The contract could embody a check for the maximum acceptable value. The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

BC - Blacklists Addresses

Criticality	Critical
Location	LinerPowerPoint.sol#L470
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `addBots` function.

```
function addBots(address[] memory _botAddress, bool _status)
external onlyOwner{
  for(uint256 i = 0; i < _botAddress.length; i++){
      botlist[_botAddress[i]] = _status;
  }
}
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

MSC - Missing Sanity Check

Criticality	Minor / Informative
Location	LinerPowerPoint.sol#L476,489
Status	Unresolved

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

The `_burnFee` nominator could exceed its denominator `_burnDivisor`, potentially causing the transfer transaction to revert.

```
function changeBurnFee(uint256 _newburnFee, uint256 _newburnDivisor)
external onlyOwner{
    _burnFee = _newburnFee;
    _burnDivisor = _newburnDivisor;
}

uint256 _burnAmount = (amount*(_burnFee))/(_burnDivisor*100);
uint256 _remainingAmount = amount - _burnAmount;
```

The variable `MinTx` could be set to values greater than the `MaxTx`.

```
function setMaxTx(uint256 _amount) external onlyOwner {
    MaxTx = _amount;
}

function setMinTx(uint256 _amount) external onlyOwner {
    MinTx = _amount;
}
```

Recommendation

The team is advised to properly check the variables according to the required specifications.

- The `_burnFee` should be lower than the `_burnDivisor`.
- The `_burnDivisor` should be greater than zero.
- The `MinTx` should be lower than the `MaxTx`.

RIS - Redundant If-Else Statement

Criticality	Minor / Informative
Location	LinerPowerPoint.sol#L720,728
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract has an if-else statement where the if branch doesn't contain any executable code. Hence, the if branch is redundant.

```
if(_isExcludedFromFee[sender] || _isExcludedFromFee[recipient]){

}else{
    require(trade, "Error: Unable to initiate your trade");
}

if(_isExcludedFromFee[sender] || _isExcludedFromFee[recipient]){

}else{
    require(!pause, "Error: Unable to initiate your transfer");
    require(amount>= MinTx, "Error: Amount not in Min Transaction
Limit");
    require(amount<= MaxTx, "Error: Amount not in Max Transaction
Limit");
}
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it. It is recommended to remove the redundant if-else statement.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	LinerPowerPoint.sol#L470,481,485,504
Status	Unresolved

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
function excludeFromFee(address account) external onlyOwner {
    _isExcludedFromFee[account] = true;
}

function includeInFee(address account) external onlyOwner {
    _isExcludedFromFee[account] = false;
}

...
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	LinerPowerPoint.sol#L446,454,462,470,476,481,485,489,493,504
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
function setAntibot() external onlyOwner {
    if(antibot){
        antibot = false;
    }else{
        antibot = true;
    }
}

function setTrade() external onlyOwner {
    if(trade){
        trade = false;
    }else{
        trade = true;
    }
}

...
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be

more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	LinerPowerPoint.sol#L218,219,235,254,423,424,425,426,466,472,485,489,494,500
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
function DOMAIN_SEPARATOR() external view returns (bytes32);
function PERMIT_TYPEHASH() external pure returns (bytes32);
function MINIMUM_LIQUIDITY() external pure returns (uint);
function WETH() external pure returns (address);
uint256 public _burnFee = 1000
uint256 public _burnDivisor = 1000
uint256 public MaxTx = 1000000 * (10 ** uint256(18))
uint256 public MinTx = 500 * (10 ** uint256(18))
address[] memory _botAddress
bool _status
uint256 _newburnDivisor
uint256 _newburnFee
uint256 _amount
address _to

...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	LinerPowerPoint.sol#L501
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
function transferForeignToken(address _token, address _to) external
onlyOwner returns(bool _sent){
    require(_token != address(this), "Can't let you take all native
token");
    uint256 _contractBalance = IBEP20(_token).balanceOf(address(this));
    _sent = IBEP20(_token).transfer(_to, _contractBalance);
}
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	transferOwnership	Public	✓	onlyOwner
IBEP20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IBEP20Metadata	Interface	IBEP20		

	name	External		-
	symbol	External		-
	decimals	External		-
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IUniswapV2Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-

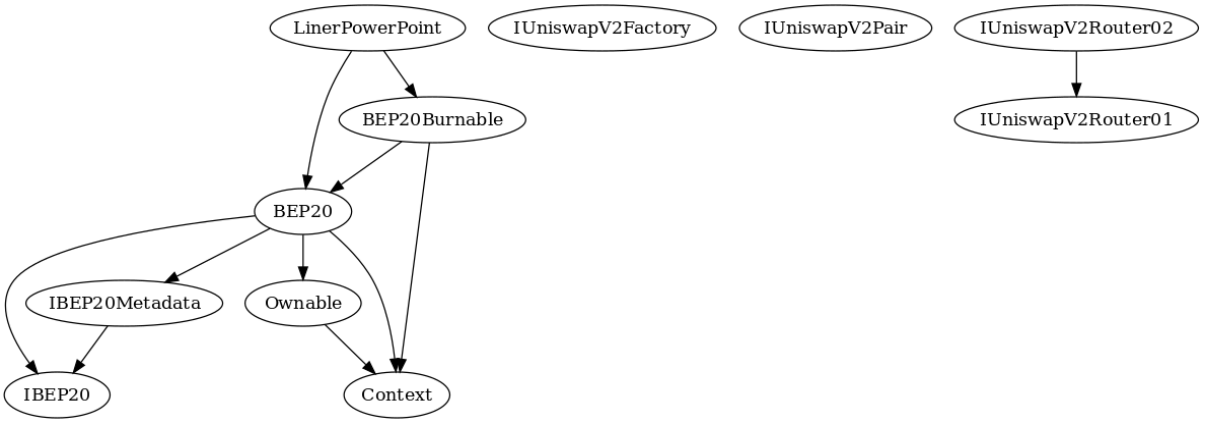
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-

	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-

	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
BEP20	Implementation	Context, IBEP20, IBEP20Metadata, Ownable		
	setAntibot	External	✓	onlyOwner
	setTrade	External	✓	onlyOwner
	setPause	External	✓	onlyOwner
	addBots	External	✓	onlyOwner
	changeBurnFee	External	✓	onlyOwner
	excludeFromFee	External	✓	onlyOwner
	includeInFee	External	✓	onlyOwner
	setMaxTx	External	✓	onlyOwner
	setMinTx	External	✓	onlyOwner
	transferForeignToken	External	✓	onlyOwner
	changeRouterVersion	External	✓	onlyOwner
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-

	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
BEP20Burnable	Implementation	Context, BEP20		
	burn	Public	✓	-
LinerPowerPoint	Implementation	BEP20, BEP20Burnable		
		Public	✓	BEP20

Inheritance Graph



Flow Graph

Summary

LINER POWER pOINT contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. There are some functions that can be abused by the owner like stopping transactions, manipulating the fees, and massively blacklist addresses. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. 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 Cyberscope's prior written consent. 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 regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document 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 report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment 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.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io