Cyberscope

Audit Report

Learn Fast Earn

January 2023

Type	BEP20
Network	BSC
Address	0xC98fbc02481E2819bbdffbaEd2d5371e03C1668c
Audited by	© cyberscope

Table of Contents

Review

Contract Name	Lefaearn
Compiler Version	v0.8.2+commit.661d1103
Optimization	200 runs
Explorer	https://bscscan.com/address/0xc98fbc02481e2819bbdffbaed2d5371e03c1668c
Address	0xc98fbc02481e2819bbdffbaed2d5371e03c1668c
Network	BSC
Symbol	LFE
Decimals	18
Total Supply	1,000,000,000

Audit Updates

Initial Audit	26 Jan 2023

Source Files

Filename	SHA256
BEP20.sol	43074670094e163ae48b7e28f8820d19cddf55e1d25ce8a9a42363f1d0bab997
BEP20Detailed.sol	4d76ac74c62c4cbe9586f91d5a61e9e227f4d7de52bb73c7d1fbbcc13e102b0b
BEPContext.sol	185ab131dbfb7cc3708c4c2ebb68626cb502b9510124ac7d3e46bb972225fa87
BEPOwnable.sol	59aec88037f20e56a5a7a8654a74aa22af9a8e0ea2f4b1d93e6e4940881d35dc
IBEP20.sol	fea064f649beb16e53397c123320dae4893c21c617173a690a1f4b205619d2dd
Lefaearn.sol	5b4234fffe8a5234d57a89397294f7ced6e9e10eceefbaa22c239e4aac3a49ac
SafeMath.sol	54c45fe70d055bf830151e348f337048c0c679d73bf7dae71b0a7c86720d5d8a

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Unresolved

BC - Blacklists Addresses

Criticality	Medium
Location	Lefaearn.sol#L41
Status	Unresolved

Description

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

```
function blackList(address _wallet, bool _status) external onlyOwner {
    isBlacklist[_wallet]= _status;
    emit changeBlacklist(_wallet, _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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L11	Unnecessary Boolean equality	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Lefaearn.sol#L22,23,24,25,26,27,41,46,51,56,61,75
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.

```
event changeBlacklist(address _wallet, bool status);
event changeCooldown(uint8 tradeCooldown);
event changeTax(uint8 _sellTax, uint8 _buyTax, uint8 _transferTax);
event changeLiquidityPoolStatus(address lpAddress, bool status);
event changeMarketingPool(address marketingPool);
event changeWhitelistTax(address _address, bool status);
bool _status
address _wallet
uint8 _tradeCooldown
address _lpAddress
address _marketingPool
uint8 _transferTax
uint8 _sellTax
uint8 _buyTax

...
```

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.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	BEP20.sol#L90,111
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _burn(address account, uint256 amount) internal {
    require(account != address(0), "BEP20: burn from the zero address");

    _balances[account] = _balances[account].sub(amount, "BEP20: burn amount exceeds
balance");
    _totalSupply = _totalSupply.sub(amount);
    emit Transfer(account, address(0), amount);
  }

...
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L11 - Unnecessary Boolean equality

Criticality	Minor / Informative
Location	Lefaearn.sol#L85,88
Status	Unresolved

Description

Boolean equality is unnecessary when comparing two boolean values. This is because a boolean value is either true or false, and there is no need to compare two values that are already known to be either true or false.

it's important to be aware of the types of variables and expressions that are being used in the contract's code, as this can affect the contract's behavior and performance. The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
liquidityPool[sender] == true
liquidityPool[receiver] == true
```

Recommendation

Using the boolean value itself is clearer and more concise, and it is generally considered good practice to avoid unnecessary boolean equalities in Solidity code.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	Lefaearn.sol#L57
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
marketingPool = _marketingPool
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	SafeMath.sol#L3 Lefaearn.sol#L2 IBEP20.sol#L3 BEPOwnable.sol#L3 BEPContext.sol#L3 BEP20Detailed.sol#L3 BEP20.sol#L3
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.2;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

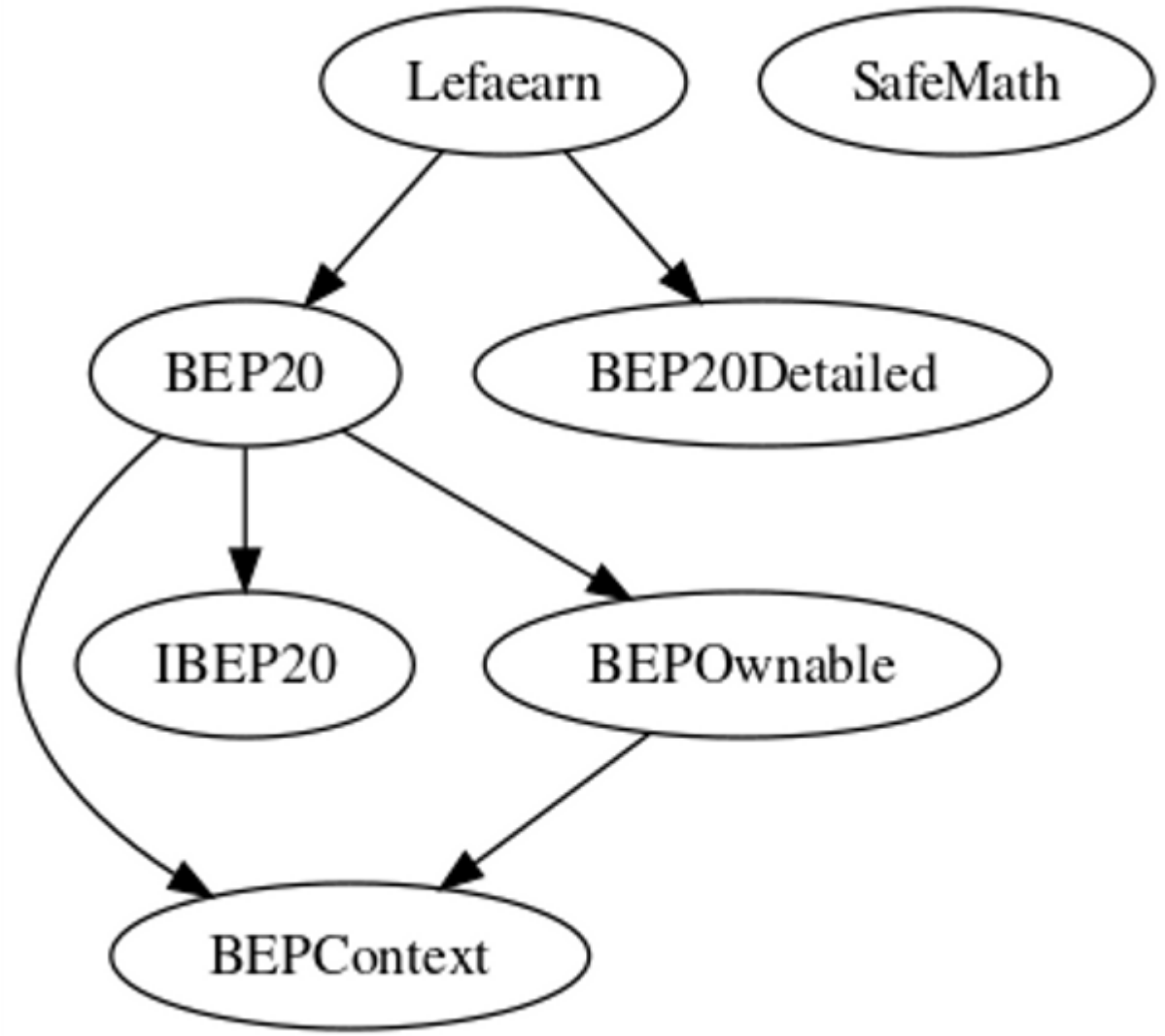
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
BEP20	Implementation	BEPContext, IBEP20, BEPOwnable		
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_burnFrom	Internal	✓	
BEP20Detailed	Implementation			
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
BEPContext	Implementation			

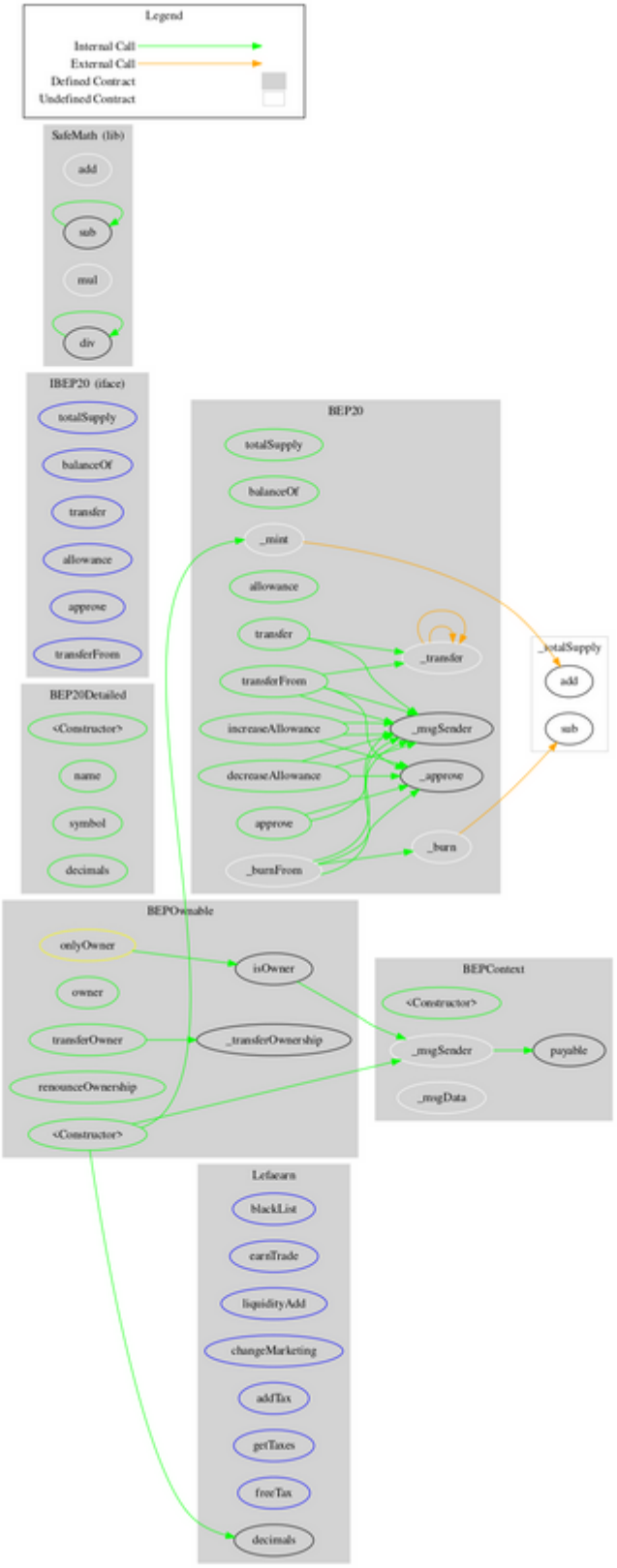
		Public	✓	-
	_msgSender	Internal		
	_msgData	Internal		
BEPOwnable	Implementation	BEPContext		
		Public	✓	-
	owner	Public		-
	isOwner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwner	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IBEP20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Lefaearn	Implementation	BEP20Detailed, BEP20		
		Public	✓	BEP20Detailed
	blackList	External	✓	onlyOwner
	earnTrade	External	✓	onlyOwner
	liquidityAdd	External	✓	onlyOwner
	changeMarketing	External	✓	onlyOwner
	addTax	External	✓	onlyOwner
	getTaxes	External		-
	freeTax	External	✓	onlyOwner

	_transfer	Internal	✓	
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
SafeMath (lib)
add
sub
mul
div
IBEP20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
BEP20Detailed
<Constructor>
name
symbol
decimals
BEP20
totalSupply
balanceOf
_mint
allowance
transfer
transferFrom
increaseAllowance
decreaseAllowance
approve
_burnFrom
_transfer
_msgSender
_approve
_burn
_totalSupply
add
sub
BEPOwnable
onlyOwner
isOwner
owner
transferOwner
_transferOwnership
renounceOwnership
<Constructor>
BEPContext
<Constructor>
_msgSender
payable
_msgData
Lefaearn
blackList
earnTrade
liquidityAdd
changeMarketing
addTax
getTaxes
freeTax
decimals

Summary

There are some functions that can be abused by the owner like 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. Additionally, it should be noted that the owner can restrict users to selling only once in a time period of up to 8.5 minutes.

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