



Cyberscope

Audit Report

Libra

February 2023

Network BSC

Address 0x0E6E4A1F02378dcA03BF3613D4517De387EFa576

Audited by © cyberscope

Table of Contents

Table of Contents	1
Review	2
Audit Updates	2
Source Files	2
Analysis	3
Diagnostics	4
CO - Code Optimization	5
Description	5
Recommendation	5
IDI - Immutable Declaration Improvement	6
Description	6
Recommendation	6
L04 - Conformance to Solidity Naming Conventions	7
Description	7
Recommendation	7
L09 - Dead Code Elimination	9
Description	9
Recommendation	9
L18 - Multiple Pragma Directives	11
Description	11
Recommendation	11
L19 - Stable Compiler Version	12
Description	12
Recommendation	12
Functions Analysis	13
Inheritance Graph	15
Flow Graph	16
Summary	17
Disclaimer	18
About Cyberscope	19

Review

Contract Name	Libra
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	200 runs
Explorer	https://bscscan.com/address/0x0e6e4a1f02378dca03bf3613d4517de387efa576
Address	0x0e6e4a1f02378dca03bf3613d4517de387efa576
Network	BSC
Symbol	LBR
Decimals	18
Total Supply	500,000

Audit Updates

Initial Audit	22 Feb 2023
Corrected Phase 2	24 Feb 2023
Corrected Phase 3	25 Feb 2023

Source Files

Filename	SHA256
Libra.sol	a86536d57aba082f05521581e400ddd8e88d11dd20d65d5041c599d343cbd6e3

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	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	Passed

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	CO	Code Optimization	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L18	Multiple Pragma Directives	Unresolved
●	L19	Stable Compiler Version	Unresolved

CO - Code Optimization

Criticality	Minor / Informative
Location	contracts/LibraToken.sol#L20,27
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 wallet address that receives the fees from the contract's transactions is used directly at the `_transfer()` function. One of the primary issues with using a hardcoded wallet address is that if the address changes, the contract will need to be updated and redeployed. This can be problematic in situations where the contract is already live on the blockchain, and users are interacting with it.

Additionally, using a hardcoded wallet address can make the contract more difficult to maintain and update. For example, if there are multiple instances in the code where the same address is used, updating the address in all instances can be a tedious and error-prone process.

```
_transfer(msg.sender, 0xdb2052dE1B1788f37E61340A2A2773bD4559C04d, taxAmount);
```

Recommendation

The team is advised to define variables for commonly used addresses, so they can be easily updated if needed.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	@openzeppelin/contracts/token/ERC20/ERC20.sol#L55,56
Status	Unresolved

Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
_nam  
_symbo
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Libra.sol#L492,494
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.

```
uint256 private constant _taxRate = 7
uint256 private constant _totalSupply = 500000 * DECIMALS
```

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	Libra.sol#L399,446
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 virtual {
    require(account != address(0), "ERC20: burn from the zero address");

    _beforeTokenTransfer(account, address(0), amount);

    uint256 accountBalance = _balances[account];
    ...
    _totalSupply -= amount;
}

emit Transfer(account, address(0), amount);

_afterTokenTransfer(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.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	Libra.sol#L4,80,106,130,489
Status	Unresolved

Description

If the contract includes multiple conflicting pragma directives, it may produce unexpected errors. To avoid this, it's important to include the correct pragma directive at the top of the contract and to ensure that it is the only pragma directive included in the contract.

```
pragma solidity ^0.8.0;  
pragma solidity ^0.8.11;
```

Recommendation

It is important to include only one pragma directive at the top of the contract and to ensure that it accurately reflects the version of Solidity that the contract is written in.

By including all required compiler options and flags in a single pragma directive, the potential conflicts could be avoided and ensure that the contract can be compiled correctly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	Libra.sol#L4,80,106,130,489
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 ^0.8.0;  
pragma solidity ^0.8.11;
```

Recommendation

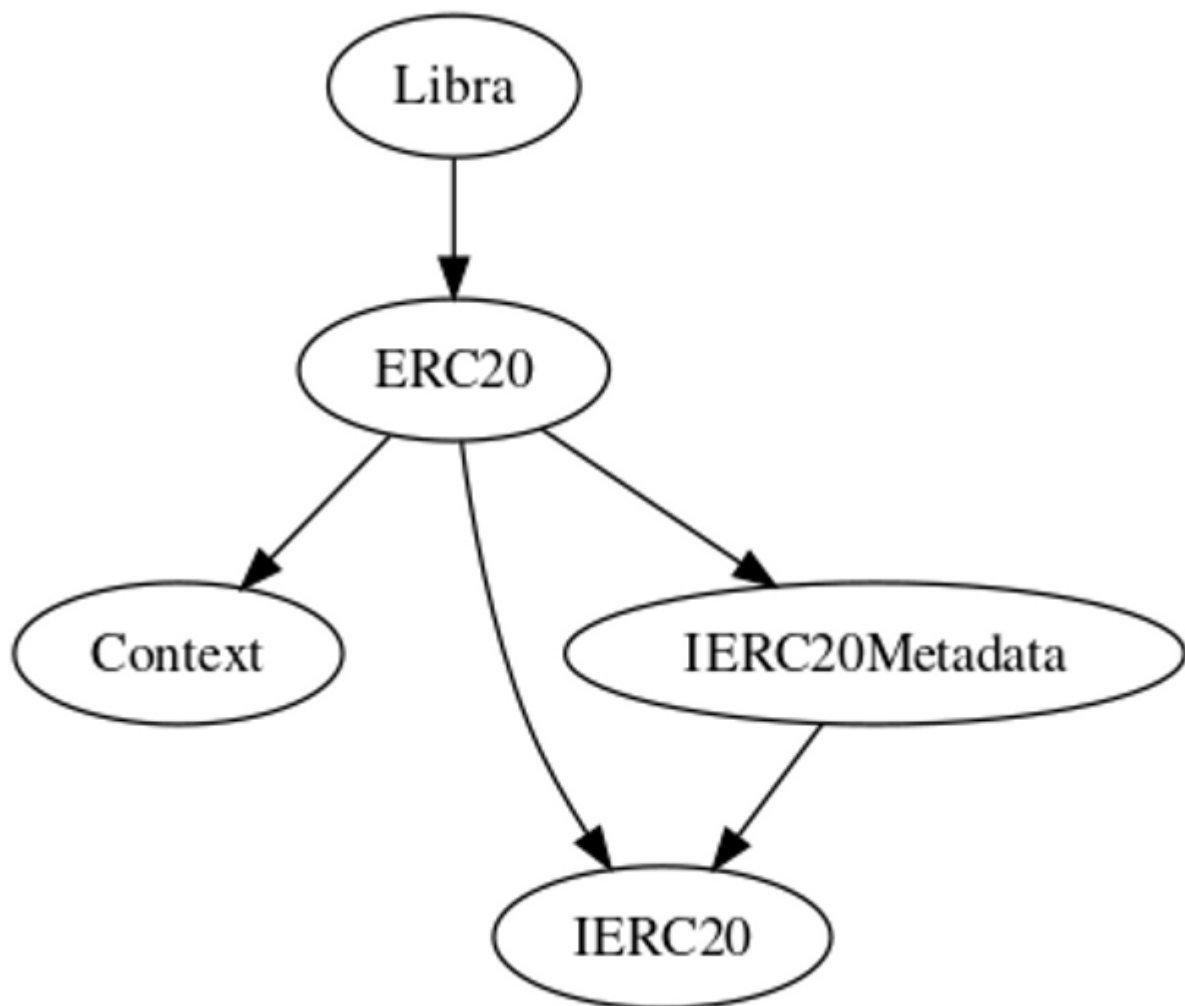
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest 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
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
ERC20	Implementation	Context, IERC20, IERC20Metadata		
		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	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
Libra	Implementation	ERC20		
		Public	✓	ERC20
	transfer	Public	✓	-
	transferFrom	Public	✓	-

Inheritance Graph



Flow Graph



Summary

Libra is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 7% fees.

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>



Cyberscope

Audit Report

Libra

February 2023

SHA256 03e965dbb2f807bf52854b65584b0fd77860382125d8942780feb2e2c9fbbe26

Audited by © cyberscope

Table of Contents

Table of Contents	1
Review	2
Audit Updates	2
Source Files	3
Analysis	4
Diagnostics	5
CO - Code Optimization	6
Description	6
Recommendation	6
AM - Allowance Misuse	7
Description	7
Recommendation	7
L04 - Conformance to Solidity Naming Conventions	8
Description	8
Recommendation	9
L19 - Stable Compiler Version	10
Description	10
Recommendation	10
Functions Analysis	11
Inheritance Graph	13
Flow Graph	14
Summary	15
Disclaimer	16
About Cyberscope	17

Review

Contract Name	Libra
Compiler Version	v0.8.11+commit.d7f03943
Optimization	200 runs
Testing Deploy	https://testnet.bscscan.com/address/0x5b7a1d062e4be63b454f356245f76976775df290
Explorer	https://testnet.bscscan.com/address/0x5b7a1d062e4be63b454f356245f76976775df290
Address	0x5b7a1d062e4be63b454f356245f76976775df290
Network	BSC_TESTNET
Symbol	LBR
Decimals	18
Total Supply	500,000

Audit Updates

Initial Audit	22 Feb 2023
----------------------	-------------

Source Files

Filename	SHA256
@openzeppelin/contracts/token/ERC20/ERC20.sol	5031430cc2613c32736d598037d3075985a2a09e61592a013dbd09a5bc2041b8
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166689e55dc037a7f2f790d057811990
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
contracts/LibraToken.sol	03e965dbb2f807bf52854b65584b0fd77860382125d8942780feb2e2c9fbbe26

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	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	Passed

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	CO	Code Optimization	Unresolved
●	AM	Allowance Misuse	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L19	Stable Compiler Version	Unresolved

CO - Code Optimization

Criticality	Minor / Informative
Location	contracts/LibraToken.sol#L20,27
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 wallet address that receives the fees from the contract's transactions is used directly at the `_transfer()` function. One of the primary issues with using a hardcoded wallet address is that if the address changes, the contract will need to be updated and redeployed. This can be problematic in situations where the contract is already live on the blockchain, and users are interacting with it.

Additionally, using a hardcoded wallet address can make the contract more difficult to maintain and update. For example, if there are multiple instances in the code where the same address is used, updating the address in all instances can be a tedious and error-prone process.

```
_transfer(msg.sender, 0xdb2052dE1B1788f37E61340A2A2773bD4559C04d, taxAmount);
```

Recommendation

The team is advised to define variables for commonly used addresses, so they can be easily updated if needed.

AM - Allowance Misuse

Criticality	Critical
Location	contracts/LibraToken.sol#L28
Status	Unresolved

Description

The contract calculates the new approved amount based on the incorrect assumption that the allowance is a balance that gets reduced by the transferred amount plus the tax amount. This assumption is incorrect because an allowance is a limit on the maximum amount that `msg.sender` can transfer on behalf of sender, and it is not a balance that gets updated with each transfer.

According to the ERC20 specification the approval amount should be calculated as the current user's allowance minus the amount specified by the `msg.sender`.

```
_approve(sender, msg.sender, allowance(sender, msg.sender) - amount + taxAmount);
```

Recommendation

The team is advised to remove the `taxAmount` from the new allowance calculation.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/LibraToken.sol#L9,11
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.

```
uint256 private constant _taxRate = 7
uint256 private constant _totalSupply = 500000 * DECIMALS
```

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>.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/LibraToken.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 ^0.8.11;
```

Recommendation

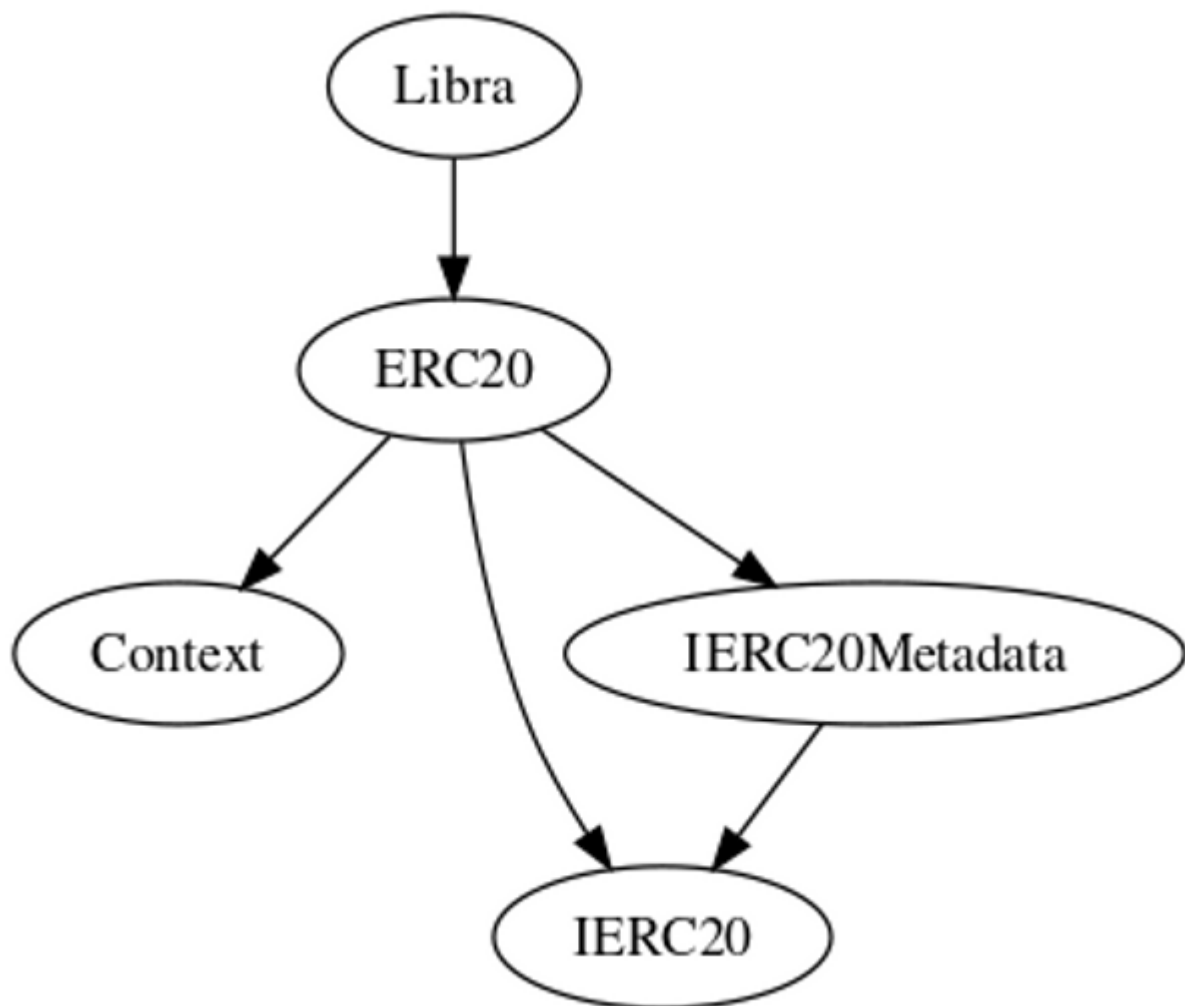
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest 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

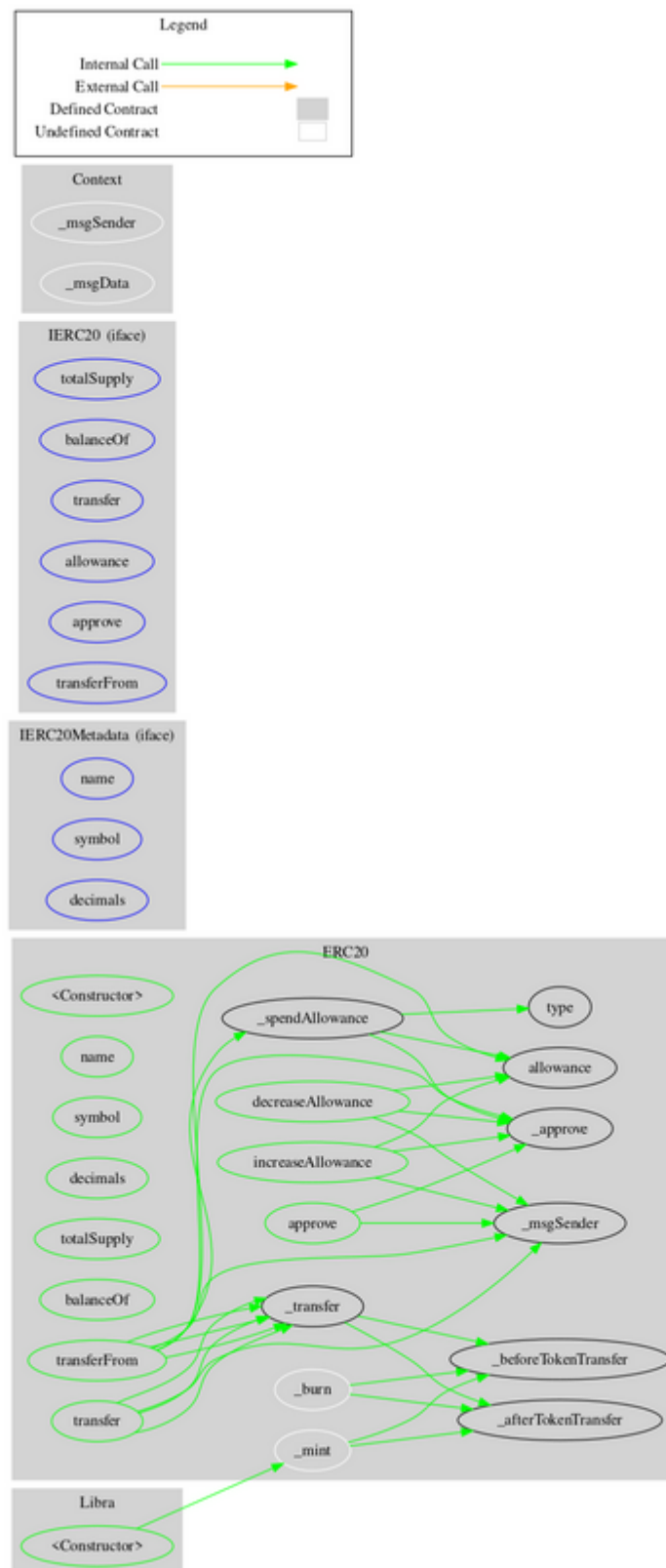
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
ERC20	Implementation	Context, IERC20, IERC20Met adata		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
IERC20Metad ata	Interface	IERC20		

	name	External		-
	symbol	External		-
	decimals	External		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Libra	Implementation	ERC20		
		Public	✓	ERC20
	transfer	Public	✓	-
	transferFrom	Public	✓	-

Inheritance Graph



Flow Graph



Summary

Libra is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors. The analysis also reported one critical issue regarding the allowance calculation, as described in detail in [AM](#) section. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 7% fees.

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>



Cyberscope

Audit Report

Libra

February 2023

SHA256 5031430cc2613c32736d598037d3075985a2a09e61592a013dbd09a5bc2041b8

Audited by © cyberscope

Table of Contents

Table of Contents	1
Review	2
Audit Updates	2
Source Files	3
Analysis	4
Diagnostics	5
CO - Code Optimization	6
Description	6
Recommendation	6
L04 - Conformance to Solidity Naming Conventions	7
Description	7
Recommendation	8
L19 - Stable Compiler Version	9
Description	9
Recommendation	9
Functions Analysis	10
Inheritance Graph	12
Flow Graph	13
Summary	14
Disclaimer	15
About Cyberscope	16

Review

Contract Name	Libra
Compiler Version	v0.8.11+commit.d7f03943
Optimization	200 runs
Testing Deploy	https://testnet.bscscan.com/address/0xf413ff91768bfc0542c3b6af377295a1b325f555
Address	0xf413ff91768bfc0542c3b6af377295a1b325f555
Network	BSC_TESTNET
Symbol	LBR
Decimals	18
Total Supply	500,000

Audit Updates

Initial Audit	22 Feb 2023 https://github.com/cyberscope-io/audits/tree/main/lbr/v1/audit.pdf
Corrected Phase 2	24 Feb 2023

Source Files

Filename	SHA256
@openzeppelin/contracts/token/ERC20/ERC20.sol	5031430cc2613c32736d598037d3075985a2a09e61592a013dbd09a5bc2041b8
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166689e55dc037a7f2f790d057811990
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
contracts/Libra.sol	a7538741dd25d898f4f177f7eb261a4df585153c634ceaa33c741b85f9cc9d45

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	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	Passed

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	CO	Code Optimization	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L19	Stable Compiler Version	Unresolved

CO - Code Optimization

Criticality	Minor / Informative
Location	contracts/LibraToken.sol#L20,27
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 wallet address that receives the fees from the contract's transactions is used directly at the `_transfer()` function. One of the primary issues with using a hardcoded wallet address is that if the address changes, the contract will need to be updated and redeployed. This can be problematic in situations where the contract is already live on the blockchain, and users are interacting with it.

Additionally, using a hardcoded wallet address can make the contract more difficult to maintain and update. For example, if there are multiple instances in the code where the same address is used, updating the address in all instances can be a tedious and error-prone process.

```
_transfer(msg.sender, 0xdb2052dE1B1788f37E61340A2A2773bD4559C04d, taxAmount);
```

Recommendation

The team is advised to define variables for commonly used addresses, so they can be easily updated if needed.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/LibraToken.sol#L9,11
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.

```
uint256 private constant _taxRate = 7
uint256 private constant _totalSupply = 500000 * DECIMALS
```

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>.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/LibraToken.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 ^0.8.11;
```

Recommendation

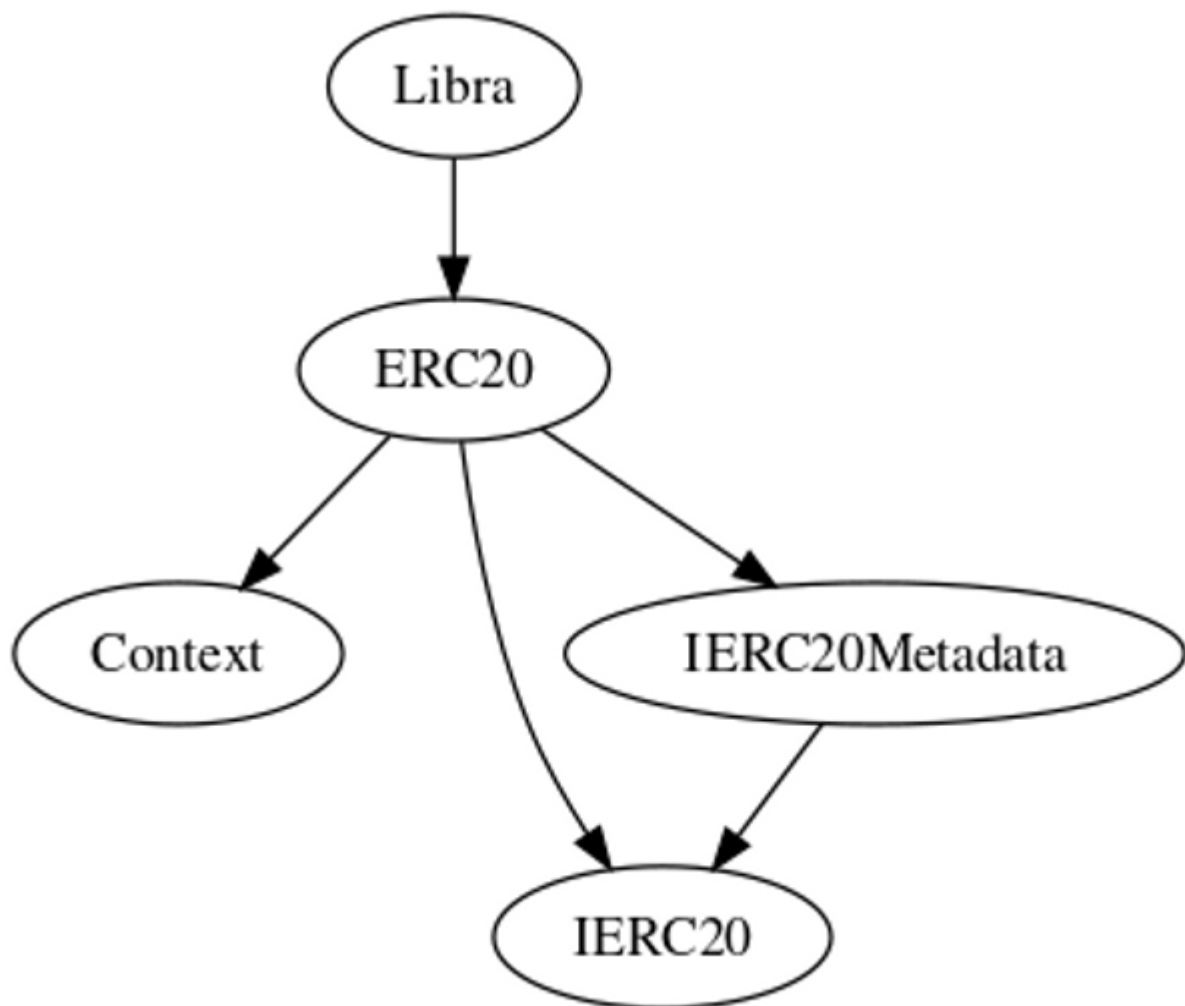
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest 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

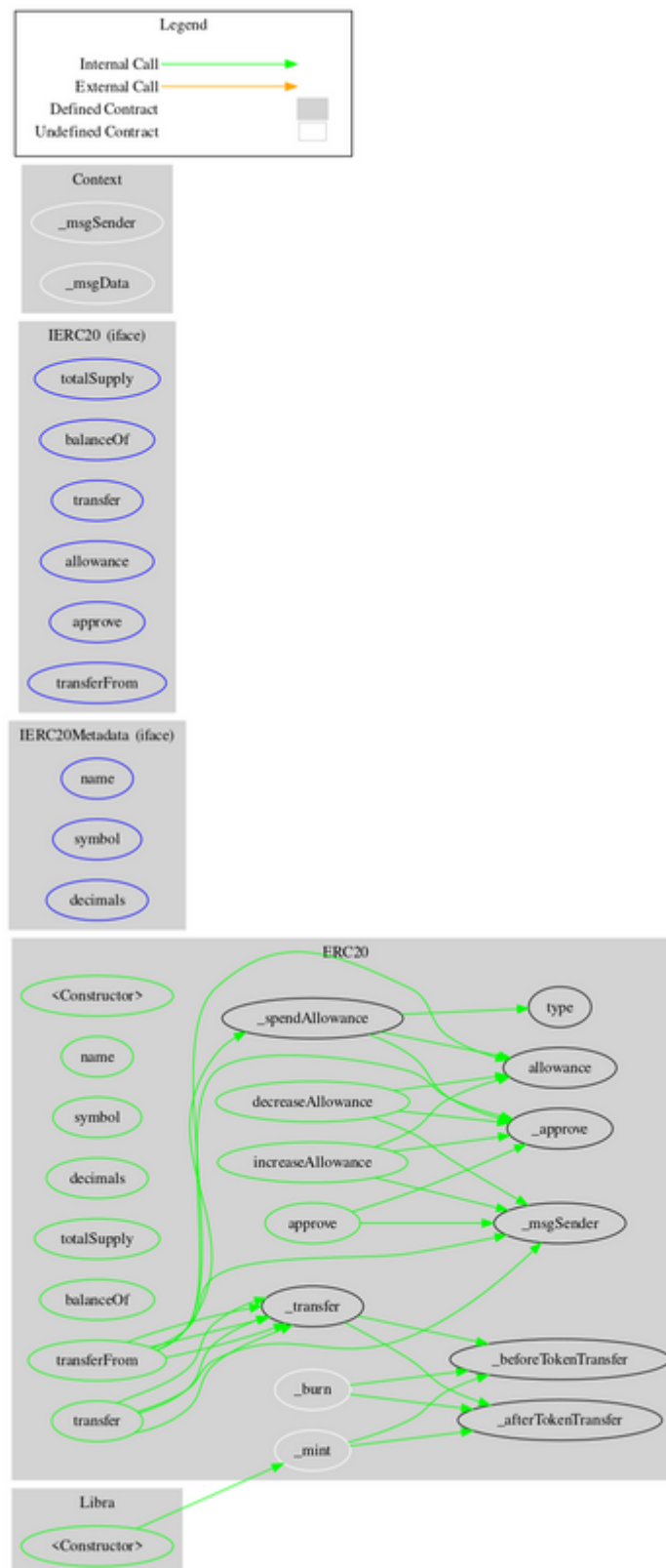
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
ERC20	Implementation	Context, IERC20, IERC20Met adata		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
IERC20Metad ata	Interface	IERC20		

	name	External		-
	symbol	External		-
	decimals	External		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Libra	Implementation	ERC20		
		Public	✓	ERC20
	transfer	Public	✓	-
	transferFrom	Public	✓	-

Inheritance Graph



Flow Graph



Summary

Libra is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 7% fees.

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>