Cyberscope

Audit Report

Lyfebloc Token

April 2023

Network	ETH
Address	0x5AC83BfbFcebb3397A40fD259dBE7a4bE04237d3
Audited by	© cyberscope

Table of Contents

Review

Contract Name	GovernanceERC20
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	2000 runs
Explorer	https://etherscan.io/address/0x5ac83bfbfcebb3397a40fd259dbe7a4be04237d3
Address	0x5ac83bfbfcebb3397a40fd259dbe7a4be04237d3
Network	ETH
Symbol	LBT
Decimals	18
Total Supply	10.000.000

Audit Updates

Initial Audit	22 Apr 2023 https://github.com/cyberscope-io/audits/blob/main/3-lbt/v1/audit.pdf
Corrected Phase 2	24 Apr 2023

Source Files

Filename	SHA256
@openzeppelin/contracts-upgradeable/governance/utils/IVotesUpgradeable.sol	400936c02700eb4147c65a91a15fb6f90d074d7519f8ebce49dce78a2c917186
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	98ce2984e449716f24043a8c11bbe969a6d34878b1d522b92c88d62708ba3376
@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol	7307fb68607d3c93995797209010e5048c9cc1777f3b97dc7940f41a7327d080
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol	6d6ffe69a38a39c69acde1dd5edb74f80cff046c4a66d1cd816b98ca741c9a43
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-IERC20PermitUpgradeable.sol	b97515a88e75c313eacf0a27c9439ef371d86d4c2730d3b13076640942f813df
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/ERC20VotesUpgradeable.sol	88763a9a0b498ca738c9a1c0c33a56464e0e8a2ad466426fe10b01cd9e01e2ae
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol	68bcca423fc72ec9625e219c9e36306c726a347e43f3711467c579bd3f6500c8
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	4e09a7479aa3e7c313f8fc141c4c8fc04e0abfeb8754615ef7d78ec94c298b07
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	1d7d481b79fd54d957c9a0696f6227f7799fec01d8ba41f5c130a7cc6b4eddc9
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	5fb301961e45cb482fe4e05646d2f529aa449fe0e90c6671475d6a32356fa2d4
@openzeppelin/contracts-upgradeable/utils/CountersUpgradeable.sol	5c1ac829a429b0c2ca9b4c9ed8b78d412320e9175e45f088c4e9056ef95fbf21

@openzeppelin/contracts-upgradeable/utils/cryptography/ECDSAUpgradeable.sol	2aee2a508bebf8e55bf78814d9d66a7a21c35c171e4010dfc3888c031f193628
@openzeppelin/contracts-upgradeable/utils/cryptography/EIP712Upgradeable.sol	91e9d20515fa1516a9e9dd754b8a3ced5552f955b039dbe69d08c566fbd2e024
@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol	fd84e5284eccc479268f0ef36b830019d4f7999ceb7959430d8d8d9e602dd4ef
@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol	a39bc026ad6214e9ecd526bd4a1ddf9862d80bd4a9d0d031d9bafa4c3c147c0b
@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol	158a0316fa289fad12c2ca764449e43e6724fb79c58fc438508d116f9af46b39
@openzeppelin/contracts-upgradeable/utils/math/SafeCastUpgradeable.sol	647d03e70d45c15cd9aa3afc3b32de945ec024a022614e263f33bb35c557ac94
@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol	68f5690fc266a6b48386c28cbfd72ec67c24b05a51ce26d24103577c15f61401
src/core/dao/IDAO.sol	1b5da3ee9786cc15cfb214d513b77fe3573a8016e44aa498cc64fddae2cc74ce
src/core/plugin/dao-authorizable/DaoAuthorizableUpgradeable.sol	d794f2cec838a02c67272fad4ac44a80535471304badb2aa8c6b38e72e13ce00
src/core/utils/auth.sol	d759140aaed34ee0cf7aa2d868a169bfb1aaaf46efc5c1c1a82637d235c4cc4b
src/token/ERC20/governance/GovernanceERC20.sol	69468ebc5a80999f8c884697d4a4a94885a97a9fd1a0842fc5a7b690db6b0118
src/token/ERC20/IERC20MintableUpgradeable.sol	86a20354d986448a7c4e0a189ac73ac9a23c1514765d109a3c78a6a2930ecf52

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	4

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	4	0	0	0

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	Unresolved
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

MT - Mints Tokens

Criticality	Minor / Informative
Location	src/token/ERC20/governance/GovernanceERC20.sol#L106
Status	Unresolved

Description

The contract's MINT_PERMISSION_ID role has the authority to mint tokens. The MINT_PERMISSION_ID role may take advantage of it by calling the `mint` function. As a result, the contract tokens will be highly inflated.

```
function mint(address to, uint256 amount) external override
auth(MINT_PERMISSION_ID) {
  _mint(to, amount);
}
```

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. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Revoking all the addresses from the MINT_PERMISSION_ID role and renouncing owership will eliminate the threats but it is non-reversible.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	L05	Unused State Variable	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved

L05 - Unused State Variable

Criticality	Minor / Informative
Location	src/core/plugin/dao-authorizable/DaoAuthorizableUpgradeable.sol#L38
Status	Unresolved

Description

An unused state variable is a state variable that is declared in the contract, but is never used in any of the contract's functions. This can happen if the state variable was originally intended to be used, but was later removed or never used.

Unused state variables can create clutter in the contract and make it more difficult to understand and maintain. They can also increase the size of the contract and the cost of deploying and interacting with it.

```
uint256[49] private __gap
```

Recommendation

To avoid creating unused state variables, it's important to carefully consider the state variables that are needed for the contract's functionality, and to remove any that are no longer needed. This can help improve the clarity and efficiency of the contract.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	src/token/ERC20/governance/GovernanceERC20.sol#L64,65,66,67,93
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.

```
IDAO _dao
string memory _name
string memory _symbol
MintSettings memory _mintSettings
bytes4 _interfaceId
```

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.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	src/token/ERC20/governance/GovernanceERC20.sol#L81
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 i
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IVotesUpgradeable	Interface			
	getVotes	External		-
	getPastVotes	External		-
	getPastTotalSupply	External		-
	delegates	External		-
	delegate	External	✓	-
	delegateBySig	External	✓	-
Initializable	Implementation			
	_disableInitializers	Internal	✓	
	_getInitializedVersion	Internal		
	_isInitializing	Internal		
ERC20Upgradeable	Implementation	Initializable, ContextUpgradeable, IERC20Upgradeable, IERC20MetadataUpgradeable		
	__ERC20_init	Internal	✓	onlyInitializing

	__ERC20_init_unchained	Internal	✓	onlyInitializing
	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	✓	
ERC20PermitUpgradeable	Implementation	Initializable, ERC20Upgradeable, IERC20PermitUpgradeable, EIP712Upgradeable		

	__ERC20Permit_init	Internal	✓	onlyInitializing
	__ERC20Permit_init_unchained	Internal	✓	onlyInitializing
	permit	Public	✓	-
	nonces	Public		-
	DOMAIN_SEPARATOR	External		-
	_useNonce	Internal	✓	
IERC20PermitUpgradeable	Interface			
	permit	External	✓	-
	nonces	External		-
	DOMAIN_SEPARATOR	External		-
ERC20VotesUpgradeable	Implementation	Initializable, IVotesUpgradeable, ERC20PermitUpgradeable		
	__ERC20Votes_init	Internal	✓	onlyInitializing
	__ERC20Votes_init_unchained	Internal	✓	onlyInitializing
	checkpoints	Public		-
	numCheckpoints	Public		-
	delegates	Public		-
	getVotes	Public		-
	getPastVotes	Public		-
	getPastTotalSupply	Public		-

	_checkpointsLookup	Private		
	delegate	Public	✓	-
	delegateBySig	Public	✓	-
	_maxSupply	Internal		
	_mint	Internal	✓	
	_burn	Internal	✓	
	_afterTokenTransfer	Internal	✓	
	_delegate	Internal	✓	
	_moveVotingPower	Private	✓	
	_writeCheckpoint	Private	✓	
	_add	Private		
	_subtract	Private		
	_unsafeAccess	Private		
IERC20MetadataUpgradeable	Interface	IERC20Upgradeable		
	name	External		-
	symbol	External		-
	decimals	External		-
IERC20Upgradeable	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-

	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
AddressUpgradeable	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	verifyCallResultFromTarget	Internal		
	verifyCallResult	Internal		
	_revert	Private		
ContextUpgradeable	Implementation	Initializable		
	__Context_init	Internal	✓	onlyInitializing
	__Context_init_unchained	Internal	✓	onlyInitializing
	_msgSender	Internal		
	_msgData	Internal		

CountersUpgradeable	Library			
	current	Internal		
	increment	Internal	✓	
	decrement	Internal	✓	
	reset	Internal	✓	
ECDSAUpgradeable	Library			
	_throwError	Private		
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	toEthSignedMessageHash	Internal		
	toEthSignedMessageHash	Internal		
	toTypedDataHash	Internal		
EIP712Upgradeable	Implementation	Initializable		
	__EIP712_init	Internal	✓	onlyInitializing
	__EIP712_init_unchained	Internal	✓	onlyInitializing
	_domainSeparatorV4	Internal		
	_buildDomainSeparator	Private		

	_hashTypedDataV4	Internal		
	_EIP712NameHash	Internal		
	_EIP712VersionHash	Internal		
ERC165Upgradeable	Implementation	Initializable, IERC165Upgradeable		
	__ERC165_init	Internal	✓	onlyInitializing
	__ERC165_init_unchained	Internal	✓	onlyInitializing
	supportsInterface	Public		-
IERC165Upgradeable	Interface			
	supportsInterface	External		-
StringsUpgradeable	Library			
	toString	Internal		
	toHexString	Internal		
	toHexString	Internal		
	toHexString	Internal		
IDAO	Interface			
	hasPermission	External		-
	setMetadata	External	✓	-
	execute	External	✓	-

	deposit	External	Payable	-
	setTrustedForwarder	External	✓	-
	getTrustedForwarder	External		-
	setSignatureValidator	External	✓	-
	isValidSignature	External	✓	-
	registerStandardCallback	External	✓	-
DaoAuthorizableUpgradeable	Implementation	ContextUpgradeable		
	__DaoAuthorizableUpgradeable_init	Internal	✓	onlyInitializing
	dao	Public		-
	_auth	Public		-
GovernanceERC20	Implementation	IERC20MintableUpgradeable, Initializable, ERC165Upgradeable, ERC20VotesUpgradeable, DaoAuthorizableUpgradeable		
		Public	✓	-
	initialize	Public	✓	initializer
	supportsInterface	Public		-
	mint	External	✓	auth
	_afterTokenTransfer	Internal	✓	

IERC20MintableUpgradeable	Interface			
	mint	External	✓	-

Summary

Lyfebloc Token contract implements a token mechanism with voting functionality. The voting power belongs to the wallet addresses that received tokens in the first transfer or mint. This audit investigates security issues, business logic concerns, and potential improvements. There are some functions that can be abused by the owner like mint tokens. if the contract owner abuses the mint functionality, then the contract will be highly inflated. 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