Cyberscope

Audit Report

KimberLite Token

February 2023

Type	BEP20
Network	BSC
Address	0x5Cb007F759d897E450D4B15Ac0EAe2001A0b35Ae
Audited by	© cyberscope

Table of Contents

Review

Contract Name	KimberLiteToken
Compiler Version	v0.8.7+commit.e28d00a7
Optimization	200 runs
Explorer	https://bscscan.com/address/0x5cb007f759d897e450d4b15ac0eae2001a0b35ae
Address	0x5cb007f759d897e450d4b15ac0eae2001a0b35ae
Network	BSC
Symbol	KIMBER
Decimals	18
Total Supply	200,000,000

Audit Updates

Initial Audit	06 Feb 2023

Source Files

Filename	SHA256
KimberLiteToken.sol	6a5b751ec68b5fe9dfe681af12abcaede7ccd5c9e9bdd0079d304a58df2874a9

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	Passed

ST - Stops Transactions

Criticality	Minor / Informative
Location	KimberLiteToken.sol#L767
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users. The owner may take advantage of it by calling the `pause()` function. As a result, all of the contract's transactions will be paused.

```
require(!paused(), "KimberLiteToken: Contract is paused");
```

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
●	L09	Dead Code Elimination	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved
●	L19	Stable Compiler Version	Unresolved

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	KimberLiteToken.sol#L633,684
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 _beforeTokenTransfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual {}

function _setupDecimals(uint8 decimals_) internal {
        _decimals = decimals_;
    }
```

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.

L15 - Local Scope Variable Shadowing

Criticality	Minor / Informative
Location	KimberLiteToken.sol#L669
Status	Unresolved

Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
string memory symbol
string memory name
```

Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	KimberLiteToken.sol#L8,34,106,197,281,309,664,693,736
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;
```

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.

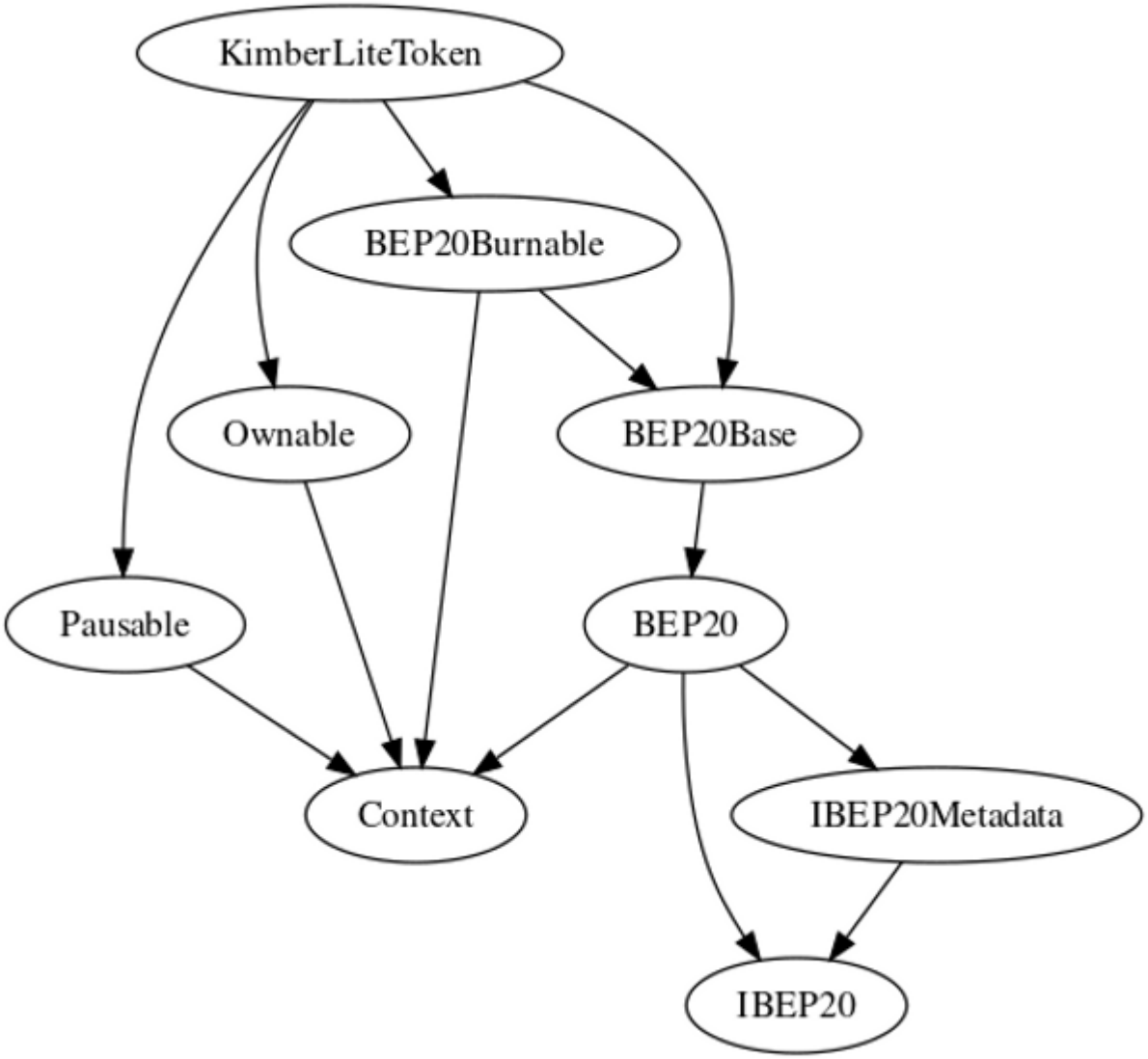
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_setOwner	Private	✓	
Pausable	Implementation	Context		
		Public	✓	-
	paused	Public		-
	_pause	Internal	✓	whenNotPaused
	_unpause	Internal	✓	whenPaused
IBEP20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-

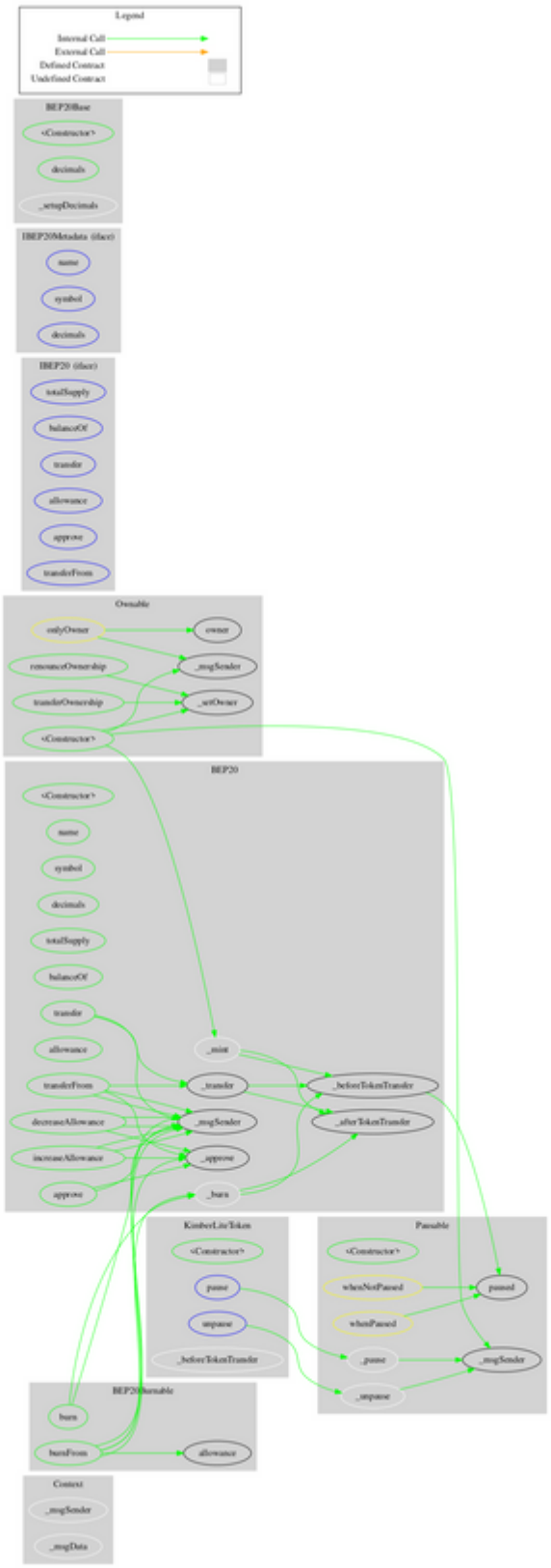
	transferFrom	External	✓	-
IBEP20Metadata	Interface	IBEP20		
	name	External		-
	symbol	External		-
	decimals	External		-
BEP20	Implementation	Context, IBEP20, IBEP20Metadata		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
BEP20Base	Implementation	BEP20		

		Public	✓	BEP20
	decimals	Public		-
	_setupDecimals	Internal	✓	
BEP20Burnable	Implementation	Context, BEP20Base		
	burn	Public	✓	-
	burnFrom	Public	✓	-
KimberLiteToken	Implementation	BEP20Base, BEP20Burnable, Ownable, Pausable		
		Public	✓	BEP20Base
	pause	External	✓	onlyOwner
	unpause	External	✓	onlyOwner
	_beforeTokenTransfer	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
BEP20Base
<Constructor>
decimals
_setupDecimals
IBEP20Metadata (iface)
name
symbol
decimals
IBEP20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Ownable
onlyOwner
owner
renounceOwnership
_msgSender
transferOwnership
_setOwner
<Constructor>
BEP20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
transferFrom
decreaseAllowance
increaseAllowance
approve
_mint
_transfer
_msgSender
_approve
_burn
_beforeTokenTransfer
_afterTokenTransfer
KimberLiteToken
<Constructor>
pause
unpause
_beforeTokenTransfer
Pausable
<Constructor>
whenNotPaused
paused
whenPaused
_pause
_msgSender
_unpause
BEP20Burnable
burn
burnFrom
allowance
Context
_msgSender
_msgData

Summary

KimberLite Token is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error 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. Additionally, it should be noted that the owner has the authority to pause and unpause the transactions at any time.

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