Cyberscope

Audit Report

IBStoken

June 2023

Network	BSC
Address	0x57D2A45653B329FAc354B04cEAd92C4db71cF09f
Audited by	© cyberscope

Table of Contents

Review

Contract Name	IBStoken
Compiler Version	v0.4.26+commit.4563c3fc
Optimization	200 runs
Explorer	https://bscscan.com/address/0x57d2a45653b329fac354b04cead92c4db71cf09f
Address	0x57d2a45653b329fac354b04cead92c4db71cf09f
Network	BSC
Symbol	IBS
Decimals	18
Total Supply	1,500,000,000

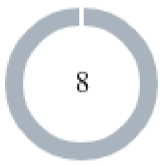
Audit Updates

Initial Audit	05 Jun 2023

Source Files

Filename	SHA256
IBStoken.sol	4cfd3c04cdf42a4703211375875ee360bf949c9cabe502c4b1c91fc32b7dda20

Findings Breakdown



●	Critical	0
●	Medium	0
●	Minor / Informative	8

Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	0	0	0	0
●	Medium	0	0	0	0
●	Minor / Informative	8	0	0	0

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	BC	Blacklists Addresses	Unresolved
●	MEM	Missing Error Messages	Unresolved
●	IBC	Invalid Blacklist Condition	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L11	Unnecessary Boolean equality	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved

ST - Stops Transactions

Criticality	Minor / Informative
Location	IBStoken.sol#L87
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.

```
function pause() onlyOwner whenNotPaused public {
  paused = true;
  emit Pause();
}
```

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

BC - Blacklists Addresses

Criticality	Minor / Informative
Location	IBStoken.sol#L222
Status	Unresolved

Description

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

```
function blackListAddress(address listAddress,  bool isBlackListed) public
whenNotPaused onlyOwner  returns (bool success) {
      return super._blackList(listAddress, isBlackListed);
}
```

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

MEM - Missing Error Messages

Criticality	Minor / Informative
Location	IBStoken.sol#L128,129,130,145,146,147,148,190,251
Status	Unresolved

Description

The contract is missing error messages for the `require` function. The `require` function is used to halt the execution of a transaction when a condition is not met, and restore the state of contract to what it was before the transaction took place. The `require` function takes a second argument which is an error message that can be used to identify the cause of the error. These error messages are absent from the contract, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

```
require(tokenBlacklist[msg.sender] == false);
require(_to != address(0));
require(_value <= balances[msg.sender]);
```

Recommendation

The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract.

IBC - Invalid Blacklist Condition

Criticality	Minor / Informative
Location	IBStoken.sol#L145
Status	Unresolved

Description

The contract implements a blacklist functionality. The conditional check used in the `transferFrom` function is not valid. The contract checks if the `msg.sender` is blacklisted, instead of checking the `_from` and `_to` addresses. The `msg.sender` could possibly be a different address than both `_from` and `_to` addresses. As a result, the contract's conditional statement is not valid.

```
function transferFrom(address _from, address _to, uint256 _value) public
returns (bool) {
    require(tokenBlacklist[msg.sender] == false);
    require(_to != address(0));
    require(_value <= balances[_from]);
    require(_value <= allowed[_from][msg.sender]);

    balances[_from] = balances[_from].sub(_value);
    balances[_to] = balances[_to].add(_value);
    allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
    emit Transfer(_from, _to, _value);
    return true;
}
```

Recommendation

The team is advised to modify the conditional statement, so that the correct addresses are checked. A recommended approach would be the following:

```
require(!tokenBlacklist[_from] && !tokenBlacklist[_to]);
```

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	IBStoken.sol#L127,140,144,158,165,170,176,202,206,210,214,218,246
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 _value
address _to
address _owner
address _from
address _spender
uint _addedValue
uint _subtractedValue
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, and 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.

L11 - Unnecessary Boolean equality

Criticality	Minor / Informative
Location	IBStoken.sol#L128,145
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.

```
require(tokenBlacklist[msg.sender] == false)
```

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	IBStoken.sol#L242
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.

```
owner = tokenOwner
```

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	IBStoken.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 ^0.4.24;
```

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.

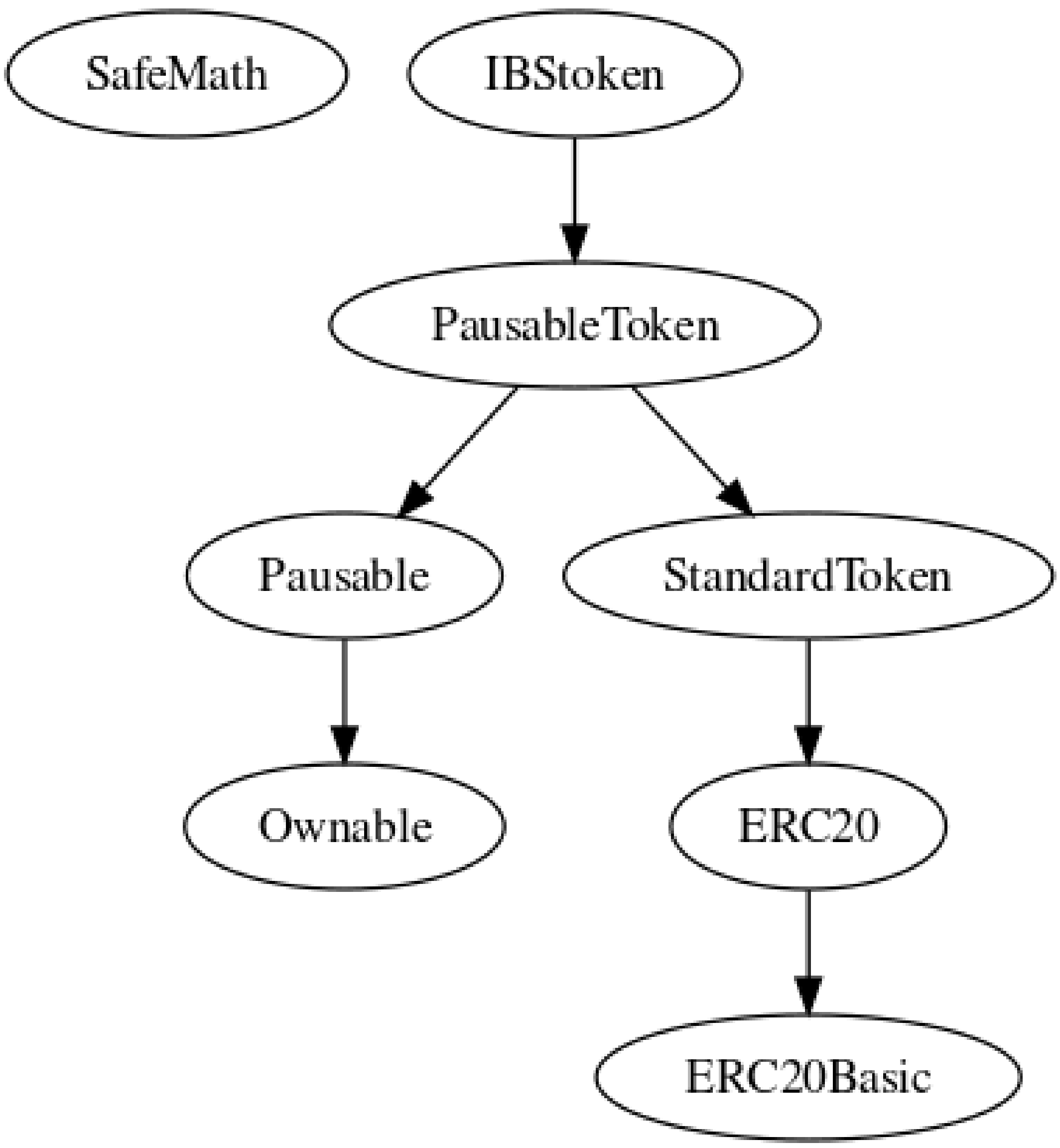
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
	mul	Internal		
	div	Internal		
	sub	Internal		
	add	Internal		
Ownable	Implementation			
	transferOwnership	Public	✓	onlyOwner
Pausable	Implementation	Ownable		
	pause	Public	✓	onlyOwner whenNotPaused
	unpause	Public	✓	onlyOwner whenPaused
ERC20Basic	Implementation			
	balanceOf	Public		-
	transfer	Public	✓	-
ERC20	Implementation	ERC20Basic		

	allowance	Public		-
	transferFrom	Public	✓	-
	approve	Public	✓	-
StandardToken	Implementation	ERC20		
	transfer	Public	✓	-
	balanceOf	Public		-
	transferFrom	Public	✓	-
	approve	Public	✓	-
	allowance	Public		-
	increaseApproval	Public	✓	-
	decreaseApproval	Public	✓	-
	_blackList	Internal	✓	
PausableToken	Implementation	StandardToken, Pausable		
	transfer	Public	✓	whenNotPaused
	transferFrom	Public	✓	whenNotPaused
	approve	Public	✓	whenNotPaused
	increaseApproval	Public	✓	whenNotPaused
	decreaseApproval	Public	✓	whenNotPaused
	blackListAddress	Public	✓	whenNotPaused onlyOwner

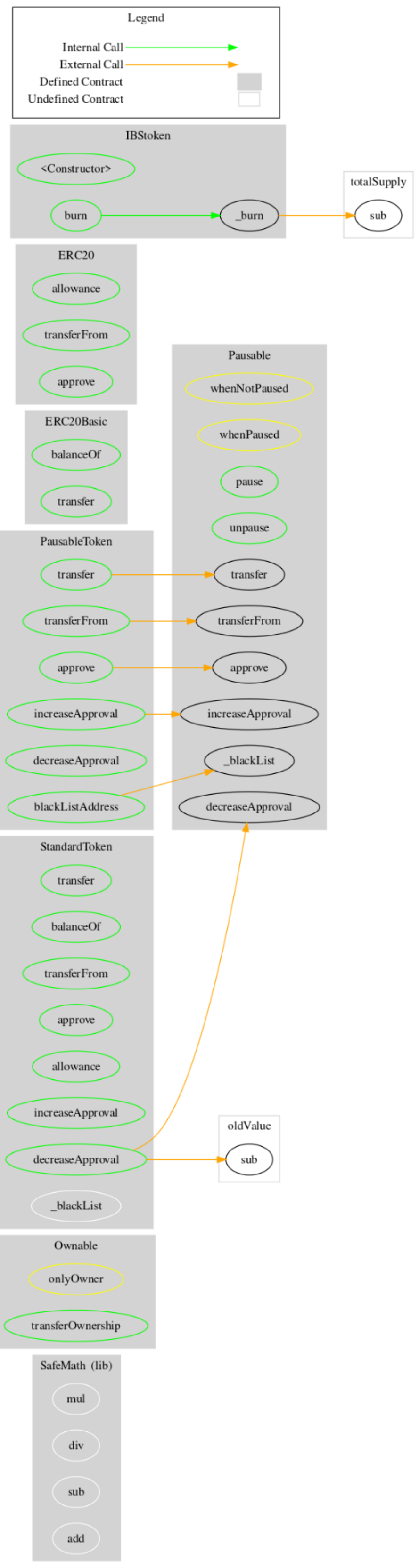
IBStoken	Implementation	PausableToken		
		Public	✓	-
	burn	Public	✓	-
	_burn	Internal	✓	

Inheritance Graph



$$\begin{tikzcd}
SafeMath & IBStoken \arrow[d] & \\
 & PausableToken \arrow[dl] \arrow[dr] & \\
Pausable \arrow[d] & & StandardToken \arrow[d] \\
Ownable & & ERC20 \arrow[d] \\
 & & ERC20Basic
\end{tikzcd}$$

Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IBStoken
<Constructor>
burn
_burn
totalSupply
sub
ERC20
allowance
transferFrom
approve
ERC20Basic
balanceOf
transfer
Pausable
whenNotPaused
whenPaused
pause
unpause
transfer
transferFrom
approve
increaseApproval
_blackList
decreaseApproval
PausableToken
transfer
transferFrom
approve
increaseApproval
decreaseApproval
blackListAddress
StandardToken
transfer
balanceOf
transferFrom
approve
allowance
increaseApproval
decreaseApproval
_blackList
oldValue
sub
Ownable
onlyOwner
transferOwnership
SafeMath (lib)
mul
div
sub
add

Summary

IBStoken contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements.

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