Cyberscope

Audit Report

Wibx

February 2023

Type	ERC20
Network	ETH
Address	0xbB97e381F1d1e94ffa2A5844F6875e6146981009
Audited by	© cyberscope

Table of Contents

Review

Contract Name	WibxToken
Compiler Version	v0.5.0+commit.1d4f565a
Optimization	200 runs
Explorer	https://etherscan.io/address/0xbb97e381f1d1e94ffa2a5844f6875e6146981009
Address	0xbb97e381f1d1e94ffa2a5844f6875e6146981009
Network	ETH
Symbol	WBX
Decimals	18
Total Supply	11,751,286,309

Audit Updates

Initial Audit	30 Jan 2023 https://github.com/cyberscope-io/audits/tree/main/wbx/v1/audit.pdf
Corrected Phase 2	02 Feb 2023

Source Files

Filename	SHA256
WibxToken.sol	6214f6b6cf601aa1d503347438bf7bcb216a112742fafde17ba3636fe828fb88

Introduction

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

The contract provides a sendBatch function, allowing users to make multiple token transfers in a single transaction, either from their own balance or with the permission of another user (through an approved allowance).

Roles

Owner

- `function renounceOwnership()`
- `function transferOwnership(address newOwner)`
- `function changeTax(uint256 amount, uint256 shift)`

Pauser

- `function addPauser(address account)`
- `function pause()`
- `function unpause()`

User

- `function transfer(address to, uint256 value)`
- `function transferFrom(address from, address to, uint256 value)`
- `function sendBatch(address[] memory recipients, uint256[] memory values, address from)`

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	Unresolved
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

ELFM - Exceeds Fees Limit

Criticality	Medium
Location	WibxToken.sol#L727
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `changeTax` function with a high percentage value.

```
function changeTax(uint256 amount, uint256 shift) public onlyOwner
{
    if (shift == 0)
    {
        require(amount <= 3, "You can't set a tax greater than 3%");
    }

    _taxContainer = TaxLib.DynamicTax(
        amount,

        // The maximum decimal places value is checked here
        TaxLib.normalizeShiftAmount(shift)
    );
}
```

Recommendation

The contract could embody a check for the maximum acceptable value. 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
●	TAI	Transfer Amount Inconsistency	Acknowledged
●	MC	Missing Check	Unresolved
●	L01	Public Function could be Declared External	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L08	Tautology or Contradiction	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved

TAI - Transfer Amount Inconsistency

Criticality	Minor / Informative
Location	WibxToken.sol#L901,998
Status	Acknowledged

Description

The `_fullTransfer()` and `transferFrom()` functions are used to transfer a specified amount of tokens to an address. The fee or tax is an amount that is charged to the sender of an ERC20 token when tokens are transferred to another address. The `taxValue` is calculated from the sender's transfer amount and forwarded to the designated `taxRecipientAddr`. The full amount from the sender is then sent to the recipient without subtracting the previously determined `taxValue`.

For better understanding, the following is a successful transaction of the contract: https://etherscan.io/tx/0x0bf4b0a12af531998204cec749aaa6138e0bececc6600078925bc14a1fed32727. To make the amounts simpler, if the transfer amount is 100 tokens and the calculated `taxValue` is 10, the recipient will receive 100 tokens while 10 tokens will be transferred to the `taxRecipientAddr`. This means the sender will be charged a total of 110 tokens, rather than just 100. If the sender's balance is insufficient, the transaction may result in an overflow and will be reverted.

```
uint256 taxValue = _applyTax(value);

// Transfer the tax to the recipient
_transfer(from, taxRecipientAddr(), taxValue);

// Transfer user's tokens
_transfer(from, to, value);
```

Recommendation

The team is encouraged to factor in the `taxValue` prior to forwarding the amount to the recipient, to ensure accuracy and prevent any potential transfer disruptions.

Team Update

The team responded with the following statement:

“The token has been created taxing the Sender instead of the Receiver for two main reasons: first, we are responsible for the GAS Fees. The sender does not need to have any amount of ETH on their wallet while using our platform and ecosystem, facilitating usage and adoption by people that don't understand a thing when dealing with Crypto. Secondly, to encourage the traditional market to use it widely as an alternative to the existing payment methods. So the reported potential issue is not really an issue but the way the Utility Token has been designed to be.”

MC - Missing Check

Criticality	Minor / Informative
Location	WibxToken.sol#L943,962
Status	Unresolved

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues. The functions `_sendBatchSelf()` and `_sendBatchFrom()` do not include any check to ensure that the msg.sender has enough balance to cover the full amount being sent to each recipient in the batch transfer. As a result, the subtraction of the amount from the sender's balance may lead to an overflow if the total amount being sent is greater than their available balance and revert the transaction. It is a best practice to always include a balance check before executing any transfer or batch transfer to prevent this type of issue.

```
function _sendBatchSelf(address[] memory recipients, uint256[] memory values, uint
transactionCount) private returns (bool)
{
    for (uint i = 0; i < transactionCount; i++)
    {
        _fullTransfer(msg.sender, recipients[i], values[i]);
    }

    return true;
}
```

Recommendation

The team is advised to properly check the variables according to the required specifications. The contract can add a balance check before each transfer in the `_sendBatchSelf()` and `_sendBatchFrom()` functions.

L01 - Public Function could be Declared External

Criticality	Minor / Informative
Location	WibxToken.sol#L914
Status	Unresolved

Description

A public function is a function that can be called from external contracts or from within the contract itself. An external function is a function that can only be called from external contracts, and cannot be called from within the contract itself.

It's generally a good idea to declare functions as external if they are only intended to be called from external contracts, as this can help to make the contract's code easier to understand and maintain. Declaring a function as external can also help to improve the contract's performance and gas consumption.

```
function sendBatch(address[] memory recipients, uint256[] memory values, address
from) public returns (bool)
    {
        /*
         * The maximum batch send should be 100 transactions.
         * Each transaction we recommend 65000 of GAS limit and the maximum block
size is 6700000.
         * 6700000 / 65000 = ~103.0769 ∴ 100 transacitons (safe rounded).
...
        {
            return _sendBatchSelf(recipients, values, transactionCount);
        }

        return _sendBatchFrom(recipients, values, from, transactionCount);
    }
```

Recommendation

It's important to choose the appropriate visibility for each function based on how it is intended to be used. Declaring a function as external when it should be public, or vice versa can lead to unnecessary gas consumption.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	WibxToken.sol#L673
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.

```
address internal _taxRecipientAddr
```

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.

L08 - Tautology or Contradiction

Criticality	Minor / Informative
Location	WibxToken.sol#L234
Status	Unresolved

Description

A tautology is a logical statement that is always true, regardless of the values of its variables. A contradiction is a logical statement that is always false, regardless of the values of its variables.

Using tautologies or contradictions can lead to unintended behavior and can make the code harder to understand and maintain. It is generally considered good practice to avoid tautologies and contradictions in the code.

```
require(shift >= 0 && shift <= 2, "You can't set more than 2 decimal places")
```

Recommendation

The team is advised to carefully consider the logical conditions is using in the code and ensure that it is well-defined and make sense in the context of the smart contract.

L15 - Local Scope Variable Shadowing

Criticality	Minor / Informative
Location	WibxToken.sol#L530,680,851
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
uint8 decimals
string memory name
address taxRecipientAddr
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	WibxToken.sol#L784
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.

```
_bchAddress = bchAddress
```

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	WibxToken.sol#L8
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 ^0.5.0;
```

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

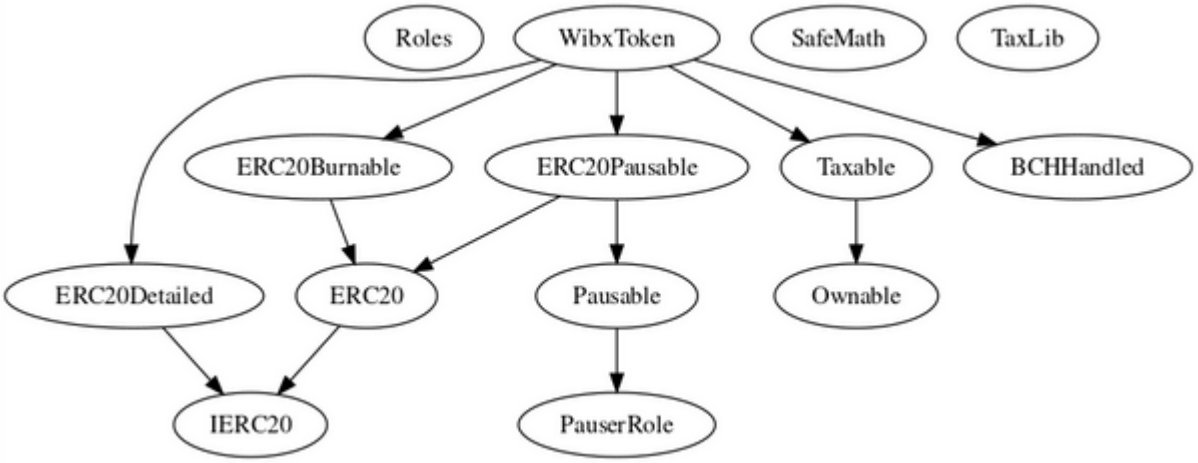
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Roles	Library			
	add	Internal	✓	
	remove	Internal	✓	
	has	Internal		
IERC20	Interface			
	transfer	External	✓	-
	approve	External	✓	-
	transferFrom	External	✓	-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
PauserRole	Implementation			
		Internal	✓	
	isPauser	Public		-
	addPauser	Public	✓	onlyPauser
	renouncePauser	Public	✓	-
	_addPauser	Internal	✓	
	_removePauser	Internal	✓	
SafeMath	Library			
	mul	Internal		
	div	Internal		

	sub	Internal		
	add	Internal		
	mod	Internal		
TaxLib	Library			
	applyTax	Internal		
	normalizeShiftAmount	Internal		
ERC20	Implementation	IERC20		
	totalSupply	Public		-
	balanceOf	Public		-
	allowance	Public		-
	transfer	Public	✓	-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_burnFrom	Internal	✓	
Pausable	Implementation	PauserRole		
		Internal	✓	
	paused	Public		-
	pause	Public	✓	onlyPauser whenNotPaused
	unpause	Public	✓	onlyPauser whenPaused

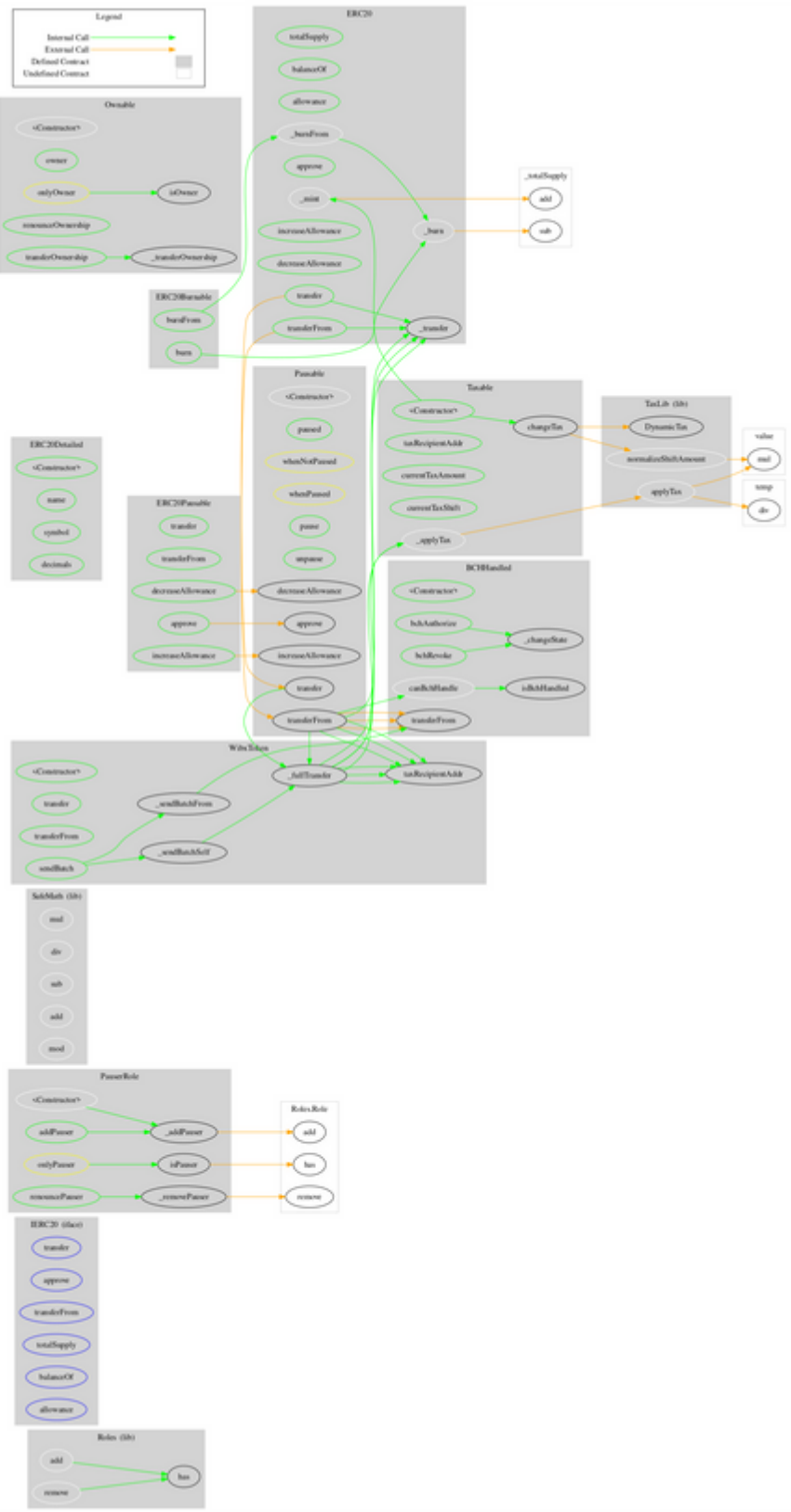
ERC20Pausable	Implementation	ERC20, Pausable		
	transfer	Public	✓	whenNotPaused
	transferFrom	Public	✓	whenNotPaused
	approve	Public	✓	whenNotPaused
	increaseAllowance	Public	✓	whenNotPaused
	decreaseAllowance	Public	✓	whenNotPaused
ERC20Detailed	Implementation	IERC20		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
ERC20Burnable	Implementation	ERC20		
	burn	Public	✓	-
	burnFrom	Public	✓	-
Ownable	Implementation			
		Internal	✓	
	owner	Public		-
	isOwner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
Taxable	Implementation	Ownable		
		Public	✓	-

	taxRecipientAddr	Public		-
	currentTaxAmount	Public		-
	currentTaxShift	Public		-
	changeTax	Public	✓	onlyOwner
	_applyTax	Internal		
BCHHandled	Implementation			
		Public	✓	-
	isBchHandled	Public		-
	bchAuthorize	Public	✓	-
	bchRevoke	Public	✓	-
	canBchHandle	Internal		
	_changeState	Private	✓	
WibxToken	Implementation	ERC20Pausable, ERC20Burnable, ERC20Detailed, Taxable, BCHHandled		
		Public	✓	ERC20Detailed BCHHandled Taxable
	transfer	Public	✓	-
	transferFrom	Public	✓	-
	sendBatch	Public	✓	-
	_sendBatchSelf	Private	✓	
	_sendBatchFrom	Private	✓	
	_fullTransfer	Private	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
ERC20
totalSupply
balanceOf
allowance
_burnFrom
approve
_mint
increaseAllowance
decreaseAllowance
transfer
transferFrom
_transfer
_burn
_totalSupply
add
sub
Ownable
<Constructor>
owner
onlyOwner
isOwner
renounceOwnership
transferOwnership
_transferOwnership
ERC20Burnable
burnFrom
burn
Pausable
<Constructor>
paused
whenNotPaused
whenPaused
pause
unpause
Taxable
<Constructor>
changeTax
taxRecipientAddr
currentTaxAmount
currentTaxShift
_applyTax
TaxLib (lib)
DynamicTax
normalizeShiftAmount
applyTax
value
mul
temp
div
ERC20Detailed
<Constructor>
name
symbol
decimals
ERC20Pausable
transfer
transferFrom
decreaseAllowance
approve
increaseAllowance
BCHHandled
<Constructor>
bchAuthorize
_changeState
bchRevoke
canBchHandle
isBchHandled
transferFrom
WibxToken
<Constructor>
transfer
transferFrom
sendBatch
_sendBatchFrom
_sendBatchSelf
_fullTransfer
taxRecipientAddr
SafeMath (lib)
mul
div
sub
add
mod
PauserRole
<Constructor>
addPauser
onlyPauser
renouncePauser
_addPauser
isPauser
_removePauser
Roles.Role
add
has
remove
IERC20 (iface)
transfer
approve
transferFrom
totalSupply
balanceOf
allowance
Roles (lib)
add
has
remove

Summary

The Wibx contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. It should be noted that the pauser has the authority to pause/unpause all transfers at any time. Additionally, it is important to note that the sender is charged the taxed amount. We state that these functionalities are necessary and required for proper protocol operations according to the business logic. Thus, we emphasize all the aspects of the contract's functionality.

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