Cyberscope

Audit Report

Blockbusters Infrastructure Service

August 2022

SHA256 5f5298c005a4ff7765732e36bfbe7b035f0335b0844dfcca5aff976ac6b45b7a

Audited by © cyberscope

Table of Contents

Contract Review

Contract Name	BlockbustersInfrastructureService
Compiler Version	v0.8.15+commit.e14f2714
Testing Deploy	https://testnet.bscscan.com/token/0xf198dE652d6Bc3d9956D2124dB523ecf13D83639
Domain	https://bbtftoken.com

Source Files

Filename	**SHA256**
contract.sol	5f5298c005a4ff7765732e36bfbe7b035f0335b0844dfcca5aff976ac6b45b7a

Audit Updates

Initial Audit	23rd August 2022
Corrected	

Introduction

The Blockbusters Infrastructure Service contract for processing the fees of a transaction. It is intended to be used by a token as an external service in order to manipulate the fees of the trade.

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	BLC	Business Logic Concern	Unresolved
●	MC	Missing Check	Unresolved
●	L01	Public Function could be Declared External	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L05	Unused State Variable	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L12	Using Variables before Declaration	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved

ST - Stops Transactions

Criticality	medium
Location	contract.sol#L295
Status	Unresolved

Description

The contract is going to revert and stop the _process transaction on every _calculateFee, if the amount is lower than the fixed fee.

```
function _process(address from_, address to_, uint256 amount_) internal virtual returns
(uint256){
        return _isServiceExempt(from_, to_) ? 0 : _calculateFee(from_, to_, amount_);
    }
```

Recommendation

A suggested implementation could apply the fees on each amount that is processed.

BLC - Business Logic Concern

Criticality	minor / informative
Location	contract.sol#L1103
Status	Unresolved

Description

The variables _txSellLimitPerHolder and _24hrSellLimitPerHolder are not initialized, they will always be zero. As a result the if statement if (_isLPPair(to_)) is not meaningful.

```
function _process(address from_, address to_, uint amount_) internal override returns (uint){
        if (_isLPPair(to_)) {
            if (!_perTxSellLimitDisabled() && amount_ > _txSellLimitPerHolder) {
                revert TXLimitExceeded();
            }
            if (!_24hrSellLimitDisabled()) {
                SellTxData storage txData = _sellTxData[from_];
                uint128 now = _timestamp();
                if (txData.timestamp == 0 || (now - txData.timestamp) > MINUTES_PER_24HRS) {
                    _sellTxData[from_] = SellTxData(uint128(amount_), now);
                } else if (txData.total + amount_ > _24hrSellLimitPerHolder) {
                    revert TXLimitExceeded();
                } else {
                    txData.total += uint128(amount_);
                }
            }
        }
        return super._process(from_, to_, amount_);
    }
```

Recommendation

The team is advised to carefully check if the implementation follows the expected logic.

MC - Missing Check

Criticality	minor / informative
Location	contract.sol#L256
Status	Unresolved

Description

The contract is processing variables that have not properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

The method _withdraw should check if there is sufficient balance before every deposit.

```
function withdraw(address to_) external virtual requires(msg.sender, _PROVIDER_FLAG() |
_NETWORK_FLAG() | _ADMIN_FLAG(), 0) {
        _withdraw(to_);
    }

function _withdraw(address to_) internal virtual {
        uint balance = address(this).balance;
        if (balance > 0) {
            address provider = _getProviderStorage();
            uint providerFee = _getProviderFeeStorage();
            if ( provider != address(0) && providerFee > 0) {
                _deposit(provider,  balance * _getProviderFeeStorage() / _PRECISION());
            }
            _deposit(to_, address(this).balance);
        }
    }
```

Recommendation

The contract should properly check the variables according to the required specifications.

L01 - Public Function could be Declared External

Criticality	minor / informative
Location	contract.sol#L381,110
Status	Unresolved

Description

Public functions that are never called by the contract should be declared external to save gas.

```
initialized
getFlags
```

Recommendation

Use the external attribute for functions never called from the contract.

L02 - State Variables could be Declared Constant

Criticality	minor / informative
Location	contract.sol#L1078,1077
Status	Unresolved

Description

Constant state variables should be declared constant to save gas.

```
_24hrSellLimitPerHolder
_txSellLimitPerHolder
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contract.sol#L165,430,15,312,410,153,181,398,418,145,1077,189,394,406,157,414,173,338,402,426,977,1078,136,169,970,1087,390,149,1055,185,161,422,321,193,177,1095
Status	Unresolved

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
_SERVICE_EXEMPT_FLAG
_REWARD_SWAP_DISABLED_FLAG
bits
_flags
_BLOCK_FROM_FLAG
_PROVIDER_FLAG
_ROUTER_FLAG
_REWARD_EXEMPT_FLAG
_PER_TX_SELL_LIMIT_DISABLED_FLAG
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions.

L05 - Unused State Variable

Criticality	minor / informative
Location	contract.sol#L1055
Status	Unresolved

Description

There are segments that contain unused state variables.

```
__gap
```

Recommendation

Remove unused state variables.

L09 - Dead Code Elimination

Criticality	minor / informative
Location	contract.sol#L181,611,446,299,450,426,201,454,582,769,592,80,924,557,917,64,355,323,193,939,644,888,881,177,185,760,189,21,430,26,898,654,60,197,232,442,434,625
Status	Unresolved

Description

Functions that are not used in the contract, and make the code's size bigger.

```
_ROUTER_FLAG
functionCallWithValue
_isRewardExempt
_receive
_isTransferLimitExempt
_REWARD_DISTRIBUTION_DISABLED_FLAG
_isServiceFeeExempt
_isRouter
functionCall
...
```

Recommendation

Remove unused functions.

L12 - Using Variables before Declaration

Criticality	minor / informative
Location	contract.sol#L857
Status	Unresolved

Description

The contract is using a variable before the declaration. This is usually happening either if it has not been declared yet or the variable has been declared in a different scope.

```
slot
```

Recommendation

The variables should be declared before any usage of them.

L14 - Uninitialized Variables in Local Scope

Criticality	minor / informative
Location	contract.sol#L857
Status	Unresolved

Description

The are variables that are defined in the local scope and are not initialized.

```
slot
```

Recommendation

All the local scoped variables should be initialized.

L15 - Local Scope Variable Shadowing

Criticality	minor / informative
Location	contract.sol#L287,286
Status	Unresolved

Description

The are variables that are defined in the local scope containing the same name from an upper scope.

```
providerFee
provider
```

Recommendation

The local variables should have different names from the upper scoped variables.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IService	Interface			
	process	External	✓	-
	withdraw	External	✓	-
	fee	External		-
	provider	External		-
	providerFee	External		-
bits	Library			
	only	Internal		
	all	Internal		
	any	Internal		
	check	Internal		
	all	Internal		
	set	Internal		
	toggle	Internal		
	isClear	Internal		
	clear	Internal		
	reset	Internal		
UsingFlags	Implementation			
	getFlags	Public		-
	_getFlags	Internal		
	_setFlags	Internal	✓	
	_getFlagStorage	Internal		
UsingPrecision	Implementation			
	_PRECISION	Internal		

UsingDefaultFlags	Implementation	UsingFlags		
	_INITIALIZED_FLAG	Internal		
	_TRANSFER_DISABLED_FLAG	Internal		
	_PROVIDER_FLAG	Internal		
	_SERVICE_FLAG	Internal		
	_NETWORK_FLAG	Internal		
	_SERVICE_EXEMPT_FLAG	Internal		
	_PROCESSING_FLAG	Internal		
	_ADMIN_FLAG	Internal		
	_BLOCKED_FLAG	Internal		
	_ROUTER_FLAG	Internal		
	_SERVICE_FEE_EXEMPT_FLAG	Internal		
	_SERVICES_DISABLED_FLAG	Internal		
	_FEE_EXEMPT_FLAG	Internal		
	_isFeeExempt	Internal		
	_isServiceFeeExempt	Internal		
	_isServiceExempt	Internal		
UsingAdmin	Implementation	UsingFlags, UsingDefaultFlags		
	_initializeAdmin	Internal	✓	
	setFlags	External	✓	requires
UsingFees	Implementation	UsingDefaultFlags, UsingPrecision		
	_setFee	Internal	✓	
	_getFee	Internal		
	_applyFee	Internal		
	_getFeesStorage	Internal		
UsingService	Implementation	IService, UsingAdmin, UsingFees		
	<Receive Ether>	External	Payable	-

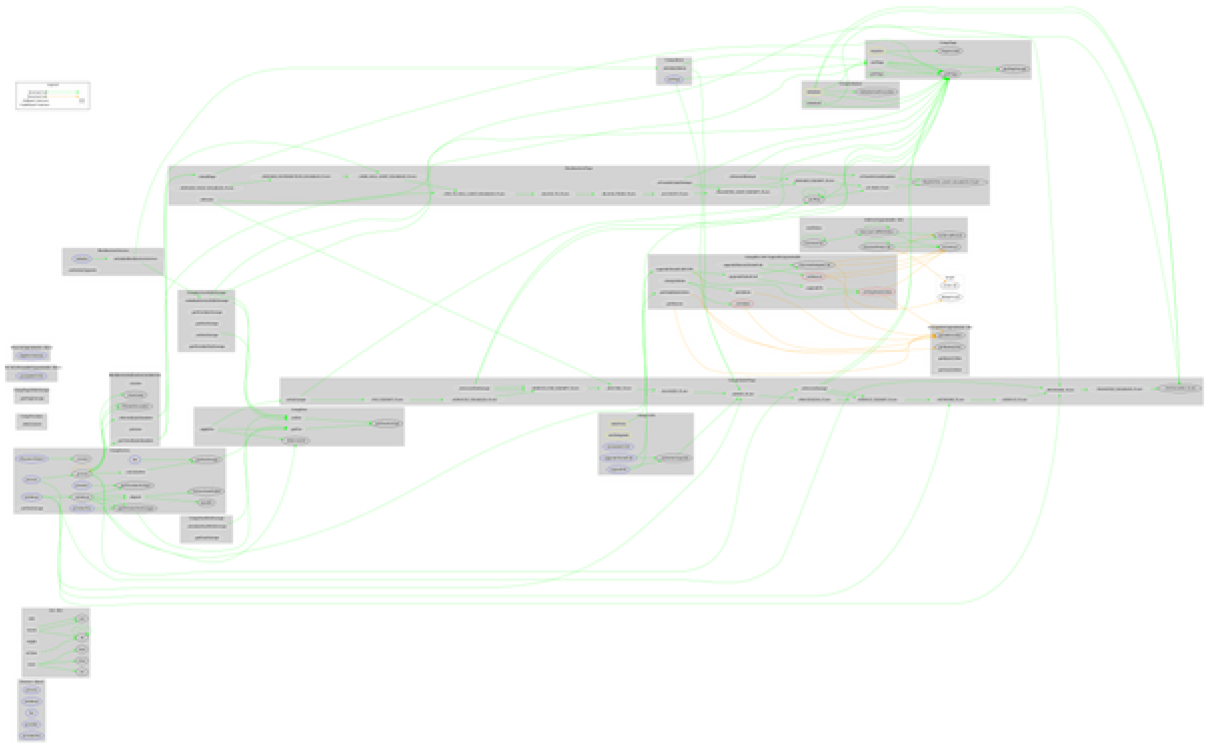
	process	External	✓	requires
	withdraw	External	✓	requires
	provider	External		-
	providerFee	External		-
	fee	External		-
	_calculateFee	Internal		
	_deposit	Internal	✓	
	_withdraw	Internal	✓	
	_process	Internal	✓	
	_receive	Internal	✓	
	_getFeeStorage	Internal		
	_setFeeStorage	Internal	✓	
	_getProviderStorage	Internal		
	_getProviderFeeStorage	Internal		
UsingFlagsWithStorage	Implementation	UsingFlags		
	_getFlagStorage	Internal		
UsingFeesWithStorage	Implementation	UsingFees		
	_initializeFeesWithStorage	Internal	✓	
	_getFeesStorage	Internal		
UsingServiceWithStorage	Implementation	UsingService, UsingFeesWithStorage		
	_initializeServiceWithStorage	Internal	✓	
	_getProviderStorage	Internal		
	_getFeeStorage	Internal		
	_setFeeStorage	Internal	✓	
	_getProviderFeeStorage	Internal		
UsingInitializer	Implementation	UsingFlags, UsingDefaultFlags		

	initialized	Public		-
BlockbustersFlags	Implementation	UsingFlags, UsingDefaultFlags, UsingAdmin		
	_TRANSFER_LIMIT_DISABLED_FLAG	Internal		
	_LP_PAIR_FLAG	Internal		
	_REWARD_EXEMPT_FLAG	Internal		
	_TRANSFER_LIMIT_EXEMPT_FLAG	Internal		
	_ACCOUNT_FLAG	Internal		
	_BLOCK_FROM_FLAG	Internal		
	_BLOCK_TO_FLAG	Internal		
	_PER_TX_SELL_LIMIT_DISABLED_FLAG	Internal		
	_24HR_SELL_LIMIT_DISABLED_FLAG	Internal		
	_REWARD_DISTRIBUTION_DISABLED_FLAG	Internal		
	_REWARD_SWAP_DISABLED_FLAG	Internal		
	_isLPPair	Internal		
	_isLPPair	Internal		
	_isTransferLimitEnabled	Internal		
	_isRewardExempt	Internal		
	_isTransferLimitExempt	Internal		
	_isRouter	Internal		
	_checkFlags	Internal		
BlockbustersFlagsWithStorage	Implementation	UsingFlagsWithStorage, BlockbustersFlags		
IERC1822ProxiableUpgradeable	Interface			
	proxiableUUID	External		-

IBeaconUpgradeable	Interface			
	implementation	External		-
AddressUpgradeable	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	verifyCallResult	Internal		
StorageSlotUpgradeable	Library			
	getAddressSlot	Internal		
	getBooleanSlot	Internal		
	getBytes32Slot	Internal		
	getUint256Slot	Internal		
UsingERC1967UpgradeUpgradeable	Implementation			
	_getImplementation	Internal		
	_setImplementation	Private	✓	
	_upgradeTo	Internal	✓	
	_upgradeToAndCall	Internal	✓	
	_upgradeToAndCallUUPS	Internal	✓	
	_getAdmin	Internal		
	_setAdmin	Private	✓	
	_changeAdmin	Internal	✓	
	_getBeacon	Internal		
	_setBeacon	Private	✓	
	_upgradeBeaconToAndCall	Internal	✓	

	_functionDelegateCall	Private	✓	
UsingUUPS	Implementation	IERC1822ProxiableUpgradeable, UsingERC1967UpgradeUpgradeable		
	proxiableUUID	External		notDelegated
	upgradeTo	External	✓	onlyProxy
	upgradeToAndCall	External	Payable	onlyProxy
	_authorizeUpgrade	Internal	✓	
BlockbustersService	Implementation	UsingServiceWithStorage, UsingInitializer, BlockbustersFlagsWithStorage, UsingUUPS		
	_initializeBlockbustersService	Internal	✓	
	initialize	External	✓	initializer
	_authorizeUpgrade	Internal	✓	requires
BlockbustersInfrastructureService	Implementation	BlockbustersService		
	_receive	Internal	✓	
	_perTxSellLimitDisabled	Internal	✓	
	_24hrSellLimitDisabled	Internal	✓	
	_timestamp	Internal		
	_process	Internal	✓	

Contract Flow



Domain Info

Domain Name	bbtftoken.com
Registry Domain ID	2685924176_DOMAIN_COM-VRSN
Creation Date	2022-03-31T18:04:42Z
Updated Date	2022-03-31T18:04:43Z
Registry Expiry Date	2023-03-31T18:04:42Z
Registrar WHOIS Server	whois.godaddy.com
Registrar URL	https://www.godaddy.com
Registrar	GoDaddy.com, LLC
Registrar IANA ID	146

The domain was created 5 months before the creation of the audit. It will expire in 7 months.

There is no public billing information, the creator is protected by the privacy settings.

Summary

This audit focuses on the business logic issues, the security concerns and the potential improvements. The owner has the authority to upgrade the contract via proxy.

Disclaimer

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Cyberscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Cyberscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Cyberscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Cyberscope team disclaims any liability for the resulting losses.

About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provide all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io