Cyberscope

Audit Report

EverSAFU Token

March 2022

Type	BEP20
Network	BSC
Address	0x40eD092304dBae1bcf1858EB24e1B14141126AcB
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	EverSAFU
Compiler Version	v0.7.6+commit.7338295f
Optimization	200 runs
Licence	Unlicense
Explorer	https://bscscan.com/token/0x40eD092304dBae1bcf1858EB24e1B14141126AcB
Symbol	EverSAFU
Decimals	5
Total Supply	325,000
Source	contract.sol
Domain	eversafu.com

Audit Updates

Initial Audit	13th March 2022
Corrected	

Contract Analysis

● Critical ● Medium ● Minor ● Pass

Severity	Code	Description
●	ST	Contract Owner is not able to stop or pause transactions
●	OCTD	Contract Owner is not able to transfer tokens from specific address
●	OTUT	Owner Transfer User's Tokens
●	ELFM	Contract Owner is not able to increase fees more than a reasonable percent (25%)
●	ULTW	Contract Owner is not able to increase the amount of liquidity taken by dev wallet more than a reasonable percent
●	MT	Contract Owner is not able to mint new tokens
●	BT	Contract Owner is not able to burn tokens from specific wallet
●	BC	Contract Owner is not able to blacklist wallets from selling

BC - Blacklisted Contracts

Criticality	medium
Location	contract.sol#L1101

Description

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

```
    require(!blacklist[sender] && !blacklist[recipient], "in_blacklist");
```

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.

OCTD - Owner Contract Tokens Drain

Criticality	minor
Location	contract.sol#L917

Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the `withdrawAllToTreasury` function.

```
    function withdrawAllToTreasury() external swapping onlyOwner {

        uint256 amountToSwap =
_gonBalances[address(this)].div(_gonsPerFragment);
        require( amountToSwap > 0,"There is no EverSAFU token deposited in token
contract");
        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = router.WETH();
        router.swapExactTokensForETHSupportingFeeOnTransferTokens(
            amountToSwap,
            0,
            path,
            treasuryReceiver,
            block.timestamp
        );
    }
```

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.

Contract Diagnostics

● Critical ● Medium ● Minor

Severity	Code	Description
●	FSA	Fixed Swap Address
●	MTS	Manipulate Total Supply
●	CO	Code Optimization
●	L01	Public Function could be Declared External
●	L02	State Variables could be Declared Constant
●	L05	Unused State Variable
●	L04	Conformance to Solidity Naming Conventions
●	L09	Dead Code Elimination
●	L07	Missing Events Arithmetic
●	L14	Uninitialized Variables in Local Scope
●	L13	Divide before Multiply Operation

FSA - Fixed Swap Address

Criticality	minor
Location	contract.sol#L623

Description

The swap address is assigned once in the constructor and it can not be changed. The decentralized swaps sometimes create a new swap version or abandon the current. A contract that cannot change the swap address may not be able to catch-up the upgrade.

```
    IPancakeSwapRouter public router;

        router =
IPancakeSwapRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
```

Recommendation

It could be better to allow the swap address mutation in case of future swap updates.

MTS - Manipulate Total Supply

Criticality	medium
Location	contract.sol#L717

Description

The total supply increased proportional to the time that has elapsed since the contract creation. This change will have a direct impact on the token price and Market Cap. This is a common feature in smart contracts called “rebase”.

```
for (uint256 i = 0; i < times; i++) {
          _totalSupply = _totalSupply
              .mul((10**RATE_DECIMALS).add(rebaseRate))
              .div(10**RATE_DECIMALS);
      }
```

Recommendation

The contract owner should carefully manage the adjustment of the circulating supply (increases or decreases), according to the token’s price fluctuations.

CO - Code Optimization

Criticality	minor
Location	contract.sol#L701

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

```
if (deltaTimeFromInit < (365 days)) {
            rebaseRate = 2355;
        } else if (deltaTimeFromInit >= (365 days)) {
            rebaseRate = 211;
        } else if (deltaTimeFromInit >= ((15 * 365 days) / 10)) {
            rebaseRate = 14;
        } else if (deltaTimeFromInit >= (7 * 365 days)) {
            rebaseRate = 2;
        }
```

Recommendation

It could remove the last 2 else if statements as they will never get executed.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L521,534,539,565,569,573,1087

Description

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

```
getLiquidityBacking
decimals
symbol
name
transferOwnership
renounceOwnership
owner
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L371,611,612,604,609,600,602,603,622,601

Description

Constant state variables should be declared constant to save gas.

```
treasuryFee
swapEnabled
sellFee
safuDividendFee
liquidityFee
feeDenominator
autofirePitFee
ZERO
DEAD
...
```

Recommendation

Add the constant attribute to state variables that never change.

L05 - Unused State Variable

Criticality	minor
Location	contract.sol#L20

Description

There are segments that contain unused state variables.

```
MAX_INT256
```

Recommendation

Remove unused state variables.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L153,154,171,191,393,347,355,966,975,1038 and 19 more

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.

```
_totalSupply
_lastAddLiquidityTime
_lastRebasedTime
_initRebaseStartTime
_autoAddLiquidity
_autoRebase
ZERO
DEAD
_isFeeExempt
...
```

Recommendation

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

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L48

Description

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

```
abs
```

Recommendation

Remove unused functions.

L07 - Missing Events Arithmetic

Criticality	minor
Location	contract.sol#L393

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
minPeriod = _minPeriod
```

Recommendation

Emit an event for critical parameter changes.

L14 - Uninitialized Variables in Local Scope

Criticality	minor
Location	contract.sol#L695

Description

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

```
rebaseRate
```

Recommendation

All the local scoped variables should be initialized.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L692,806,1087

Description

Performing divisions before multiplications may cause lose of prediction.

```
liquidityBalance = _gonBalances[pair].div(_gonsPerFragment)
_gonBalances[autoLiquidityReceiver] =
_gonBalances[autoLiquidityReceiver].add(gonAmount.div(feeDenominator).mul(liqui
dityFee))
_gonBalances[address(this)] =
_gonBalances[address(this)].add(gonAmount.div(feeDenominator).mul(_treasuryFee.
add(safuDividendFee)))
_gonBalances[autofirePit] =
_gonBalances[autofirePit].add(gonAmount.div(feeDenominator).mul(autofirePitFee)
)
feeAmount = gonAmount.div(feeDenominator).mul(_totalFee)
times = deltaTime.div(900)
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SafeMathInt	Library			
	mul	Internal		
	div	Internal		
	sub	Internal		
	add	Internal		
	abs	Internal		
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	transfer	External	✓	-
	approve	External	✓	-
	transferFrom	External	✓	-
IPancakeSwap Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-

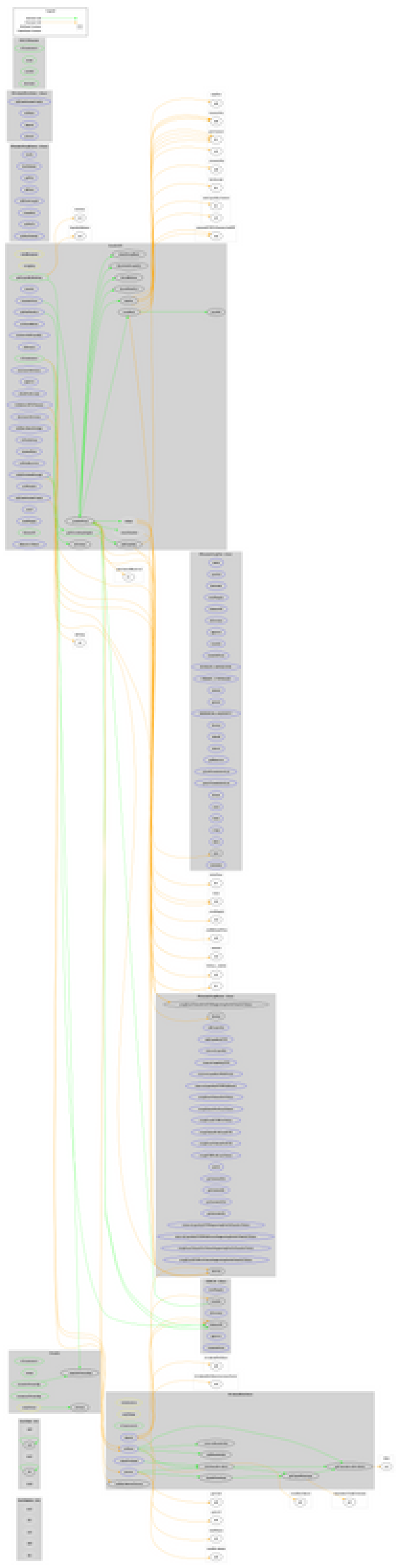
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
IPancakeSwap Router	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-

	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IPancakeSwapFactory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-
	deposit	External	Payable	-
	process	External	✓	-

DividendDistributor	Implementation	IDividendDistributor		
	<Constructor>	Public	✓	-
	setDistributionCriteria	External	✓	onlyToken
	setShare	External	✓	onlyToken
	deposit	External	Payable	onlyToken
	process	External	✓	onlyToken
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	
	claimDividend	External	✓	-
	getUnpaidEarnings	Public		-
	getCumulativeDividends	Internal		
	addShareholder	Internal	✓	
	removeShareholder	Internal	✓	
Ownable	Implementation			
	<Constructor>	Public	✓	-
	owner	Public		-
	isOwner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
ERC20Detailed	Implementation	IERC20		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
EverSAFU	Implementation	ERC20Detailed, Ownable		
	<Constructor>	Public	✓	ERC20Detailed Ownable
	rebase	Internal	✓	
	transfer	External	✓	validRecipient

	transferFrom	External	✓	validRecipient
	_basicTransfer	Internal	✓	
	_transferFrom	Internal	✓	
	takeFee	Internal	✓	
	addLiquidity	Internal	✓	swapping
	swapBack	Internal	✓	swapping
	withdrawAllToTreasury	External	✓	swapping onlyOwner
	shouldTakeFee	Internal		
	shouldRebase	Internal		
	shouldAddLiquidity	Internal		
	shouldSwapBack	Internal		
	setAutoRebase	External	✓	onlyOwner
	setAutoAddLiquidity	External	✓	onlyOwner
	allowance	External		-
	decreaseAllowance	External	✓	-
	increaseAllowance	External	✓	-
	approve	External	✓	-
	checkFeeExempt	External		-
	setIsDividendExempt	External	✓	onlyOwner
	setDistributionCriteria	External	✓	onlyOwner
	setDistributorSettings	External	✓	onlyOwner
	getCirculatingSupply	Public		-
	isNotInSwap	External		-
	manualSync	External	✓	-
	setFeeReceivers	External	✓	onlyOwner
	getLiquidityBacking	Public		-
	setWhitelist	External	✓	onlyOwner
	setBotBlacklist	External	✓	onlyOwner
	setLP	External	✓	onlyOwner
	totalSupply	External		-
	balanceOf	Public		-
	isContract	Internal		
	<Receive Ether>	External	Payable	-

Contract Flow



Summary

EverSAFU Token is an interesting project that has a friendly and growing community. The contract implements an auto-rebase mechanism, based on time passed from initialisation. There are some functions that can be abused by the owner, blacklisting wallets from transactions or allowing the contract owner to withdraw the contract balance. 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

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 provides all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io