Audit Report

Evocardano Plus

April 2023

Network	BSC
Address	0x70f1666484eE1Fce87A71Dc238565c23FFC9a91C
Audited by	© cyberscope

Table of Contents

Review

Contract Name	EvocPlus
Compiler Version	v0.7.6+commit.7338295f
Optimization	200 runs
Explorer	https://bscscan.com/address/0x70f1666484ee1fce87a71dc238565c23ffc9a91c
Address	0x70f1666484ee1fce87a71dc238565c23ffc9a91c
Network	BSC
Symbol	EVOC+
Decimals	18
Total Supply	3.000.000.000

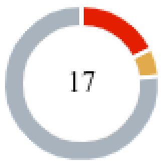
Audit Updates

Initial Audit	25 Apr 2023

Source Files

Filename	SHA256
EvocPlus.sol	9a911532a5677b4c421fc7dac5a0b85beb869913a2701d96dfd3d22805794820

Findings Breakdown



●	Critical	3
●	Medium	1
●	Minor / Informative	13

Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	0	3	0	0
●	Medium	1	0	0	0
●	Minor / Informative	13	0	0	0

Analysis

🔴 Critical 🟠 Medium ⚪ Minor / Informative 🔵 Pass

Severity	Code	Description	Status
🔴	ST	Stops Transactions	Acknowledged
🔵	OCTD	Transfers Contract's Tokens	Passed
🔵	OTUT	Transfers User's Tokens	Passed
🔴	ELFM	Exceeds Fees Limit	Acknowledged
⚪	ULTW	Transfers Liquidity to Team Wallet	Unresolved
🔴	MT	Mints Tokens	Acknowledged
🔵	BT	Burns Tokens	Passed
🟠	BC	Blacklists Addresses	Unresolved

ST - Stops Transactions

Criticality	Critical
Location	EvocPlus.sol#L741,689,692,700
Status	Acknowledged

Description

The contract owner has the authority to stop the transactions for all users excluding the owner. The owner may take advantage of it by setting the `_maxTxAmount` to zero.

```
function checkTxLimit(address sender, address recipient,
uint256 amount) internal view {
  if (recipient == pair && !isTxLimitExempt[sender]){
    require(amount <= _maxTxAmount, "Exceeded the maximum
transaction limit");
  }
}
```

The contract owner has the authority to pause or unpause the transactions for all users excluding the owner.

```
if(!authorizations[sender] && !authorizations[recipient]){
  require(!pausedToken,"Token is paused");
}
```

The contract owner has the authority to stop the transactions for all users. The owner may take advantage of it by setting the `_maxWalletToken` to zero.

```
if (!authorizations[sender] && recipient != address(this) &&
recipient != address(DEAD) && recipient != pair
  && recipient != marketingFeeReceiver && recipient !=
autoLiquidityReceiver){
  uint256 heldTokens = balanceOf(recipient);
  require((heldTokens + amount) <= _maxWalletToken,"Total
Holding is currently limited, you can not buy that much.");
}
```

The contract owner has the authority to stop the transactions for all users excluding the owner. The owner may take advantage of it by setting the `cooldownTimer` to a big number. As a result, the contract may operate as a honeypot.

```
if (recipient == pair && buyCooldownEnabled &&
!isTimelockExempt[sender]) {
  require(cooldownTimer[sender] < block.number,"No consecutive
sells allowed. Please wait.");
  cooldownTimer[sender] = block.number + cooldownTimerInterval;
}
```

Recommendation

The contract could embody a check for not allowing setting the _maxTxAmount less than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply. 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.

Team Update

The team responded with the following statement:

“We added this feature because it is necessary to pause the trade before completing our presale. Our project has gone through a hard fork, a contract update and all previous investors have already received their coins. This function was added precisely to prevent other people from adding liquidity to our contract before the token launch, as they already have the new coins in their wallets, avoiding harming our official launch!!”

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	EvocPlus.sol#L907,912,923
Status	Acknowledged

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 `setTransferBuyFee` , `setFeeDistribution` , and `setFeeSwap` functions with a high percentage value.

```
function setTransferBuyFee(uint256 _transferTaxRate, uint256 _buyTaxRate)
external onlyOwner {
  transferFee = _transferTaxRate;
  totalBuyFee = _buyTaxRate;

}
function setFeeDistribution(uint256 _liquidityFee, uint256
_reflectionFee, uint256 _marketingFee,  uint256 _burnFee, uint256
_buybackSellFee, uint256 _feeDenominator) external onlyOwner {
  liquidityFee = _liquidityFee;
  reflectionFee = _reflectionFee;
  marketingFee = _marketingFee;
  burnFee = _burnFee;
  buybackSellFee = _buybackSellFee;
  totalFee =
_liquidityFee.add(_reflectionFee).add(_marketingFee).add(_burnFee).add(_b
uybackSellFee);
  feeDenominator = _feeDenominator;
  require(totalFee <= feeDenominator);
}

function setFeeSwap(uint256 _totalBuyFee, uint256 _totalSwapFee) external
onlyOwner {
  totalBuyFee = _totalBuyFee;
  totalSwapFee = _totalSwapFee;
}
```

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. 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.

Team Update

The team responded with the following statement:

"We added this feature because it is necessary as the project has reflections, and we need to find a middle ground in fees. So that investors can receive good reflections without impacting our chart in reflection payments before the renouncement of our contract."

ULTW - Transfers Liquidity to Team Wallet

Criticality	Minor / Informative
Location	EvocPlus.sol#L813
Status	Unresolved

Description

The contract owner has the authority to set the `totalFee` to zero. As a result, the function `swapBack` will swap the contract's tokens to the native currency and send the swapped amount to the `marketingFeeReceiver` .

```
function swapBack() internal swapping {
  address[] memory path = new address[](2);
  path[0] = address(this);
  path[1] = WBNB;
  uint256 balanceBefore = address(this).balance;
  uint256 amountBNB;
  if (totalFee > 0) {
     ...
  } else {
    router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        balanceOf(address(this)),
        0,
        path,
        address(this),
        block.timestamp
    );
    amountBNB = address(this).balance.sub(balanceBefore);
      (bool tmpSuccess,) = payable(marketingFeeReceiver).call{value:
amountBNB, gas: 30000}("");
    tmpSuccess = false;
  }
}
```

Recommendation

The contract could embody a check for the maximum amount of funds that can be swapped, since a huge amount may volatile the token's price. 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.

MT - Mints Tokens

Criticality	Critical
Location	EvocPlus.sol#L657
Status	Acknowledged

Description

The minter role has the authority to mint tokens up to 10000000000 tokens which are 330% more than the initial supply. The owner may take advantage of it by calling the `mint` function. As a result, the contract tokens will be highly inflated.

```
function mint(uint256 amount) public onlyMinter returns (bool) {
  _mint( msg.sender, amount);
  return true;
}
```

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.

Team Update

The team responded with the following statement:

"We added this feature because it is necessary for our project. We will have stakes and farms in our DEX (decentralized exchange), and the contract will need to mint tokens to make payments to those who are staking."

BC - Blacklists Addresses

Criticality	Medium
Location	EvocPlus.sol#L630
Status	Unresolved

Description

The contract's authorized users have the authority to stop addresses from transactions. The owner may take advantage of it by calling the `devListAddress` function.

```
function devListAddress(address account, bool value) external
authorized{
    _isDevlisted[account] = value;
}
```

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.

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	ZD	Zero Division	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	MRM	Missing Revert Messages	Unresolved
●	DDP	Decimal Division Precision	Unresolved
●	RSK	Redundant Storage Keyword	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

ZD - Zero Division

Criticality	Minor / Informative
Location	EvocPlus.sol#L758
Status	Unresolved

Description

The contract is using variables that may be set to zero as denominators. This can lead to unpredictable and potentially harmful results, such as a transaction revert.

The variable `feeDenominator` could be set to zero address. As a result transfer transactions might revert.

```
function takeFee(address sender, address recipient, uint256
amount) internal returns (uint256) {
    uint256 feeAmount = 0;
    if (recipient == pair && totalFee > 0) {
        feeAmount = amount.mul(totalFee).div(feeDenominator);
        _balances[address(this)] =
_balances[address(this)].add(feeAmount);
        emit Transfer(sender, address(this), feeAmount);
    } else  if (sender == pair && totalBuyFee > 0) {
        feeAmount =
amount.mul(totalBuyFee).div(feeDenominator);
        _balances[address(this)] =
_balances[address(this)].add(feeAmount);
        emit Transfer(sender, address(this), feeAmount);
    } else if (recipient == pairSwap && totalSwapFee > 0) {
        feeAmount =
amount.mul(totalSwapFee).div(feeDenominator);
        _balances[address(this)] =
_balances[address(this)].add(feeAmount);
        emit Transfer(sender, address(this), feeAmount);
    } else  if (sender == pairSwap && totalBuySwapFee > 0) {
        feeAmount =
amount.mul(totalBuySwapFee).div(feeDenominator);
        _balances[address(this)] =
_balances[address(this)].add(feeAmount);
        emit Transfer(sender, address(this), feeAmount);
    }else{
        feeAmount =
amount.mul(transferFee).div(feeDenominator);
        _balances[address(this)] =
_balances[address(this)].add(feeAmount);
        emit Transfer(sender, address(this), feeAmount);
    }
    return amount.sub(feeAmount);
}
```

Recommendation

It is important to handle division by zero appropriately in the code to avoid unintended behavior and to ensure the reliability and safety of the contract. The contract should ensure that the divisor is always non-zero before performing a division operation. It should prevent the variables to be set to zero, or should not allow executing of the corresponding statements.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	EvocPlus.sol#L939
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapThreshold` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
function setSwapBackSettings(bool _enabled, uint256 _amount)
external onlyOwner {
    swapEnabled = _enabled;
    swapThreshold = _amount;
}
```

Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the total supply. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

MRM - Missing Revert Messages

Criticality	Minor / Informative
Location	EvocPlus.sol#L355,359,671,882,916,950
Status	Unresolved

Description

The contract is missing error messages. These missing error messages are 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(!initialized);
require(msg.sender == _token); _;
require(
_rewardTokenAddress != DEAD
&& _rewardTokenAddress != pair
&& _rewardTokenAddress != owner
&& _rewardTokenAddress != address(this)
);
require(holder != address(this) && holder != pair);
require(totalFee <= feeDenominator);
require(gas < 750000);
```

Recommendation

The team is advised to carefully review the source code in order to address these issues. To accelerate the debugging process and mitigate these issues, the team should use more specific and descriptive error messages.

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	EvocPlus.sol#L832
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

The variable `amountBNB` will not be splitted as expected.

```
amountBNB = address(this).balance.sub(balanceBefore);
uint256 totalBNBFee = totalFee.sub(dynamicLiquidityFee.div(2));
uint256 amountBNBLiquidity =
amountBNB.mul(dynamicLiquidityFee).div(totalBNBFee).div(2);
uint256 amountBNBReflection =
amountBNB.mul(reflectionFee).div(totalBNBFee);
uint256 amountBNBMarketing =
amountBNB.mul(marketingFee).div(totalBNBFee);
uint256 buybackTokens =

amountBNB.mul(buybackSellFee).div(totalBNBFee);
```

Recommendation

The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

RSK - Redundant Storage Keyword

Criticality	Minor / Informative
Location	EvocPlus.sol#L111
Status	Unresolved

Description

The contract uses the `storage` keyword in a view function. The `storage` keyword is used to persist data on the contract's storage. View functions are functions that do not modify the state of the contract and do not perform any actions that cost gas (such as sending a transaction). As a result, the use of the `storage` keyword in view functions is redundant.

```
Role storage role
```

Recommendation

It is generally considered good practice to avoid using the `storage` keyword in view functions because it is unnecessary and can make the code less readable.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	EvocPlus.sol#L580,585
Status	Unresolved

Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
router
distributor
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	EvocPlus.sol#L344,505,506,507,508,515,537,560
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
uint256 public dividendsPerShareAccuracyFactor = 10 ** 36
address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
address DEAD = 0x000000000000000000000000000000000000dEaD
address ZERO = 0x0000000000000000000000000000000000000000
address MKT  = 0xD10D2D5f60C81300a1d914C52594339De0BDE13A
uint256 _MaxSupply = 10000000000* (10**_decimals)
uint256 public totalBuySwapFee       = 8
uint256 public launchedAt
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	EvocPlus.sol#L243,266,322,330,331,371,392,397,401,504,505,506,507,508,510,511,514,515,518,521,523,524,526,670,793,797,804,903,908,919,924,930,935,940,945
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.

```
uint32 _pass
function WETH() external pure returns (address);
address _token
IBEP20 REWARD = IBEP20(0)
address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
uint256 _minDistribution
uint256 _minPeriod
address _address
address public REWARD =
0x3EE2200Efb3400fAbB9AacF31297cBdD1d435D47
address DEAD = 0x000000000000000000000000000000000000dEaD
address ZERO = 0x0000000000000000000000000000000000000000
address MKT  = 0xD10D2D5f60C81300a1d914C52594339De0BDE13A
string constant _name = "Evocardano Plus"
string constant _symbol = "EVOC+"

...
```

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.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	EvocPlus.sol#L372,878,904,909,920,941
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
minPeriod = _minPeriod
_maxTxAmount = maxTxAmount
transferFee = _transferTaxRate
liquidityFee = _liquidityFee
totalBuyFee = _totalBuyFee
targetLiquidity = _target
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	EvocPlus.sol#L402,925,926,927,931,932
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.

```
WBNB = _address
autoLiquidityReceiver = _autoLiquidityReceiver
marketingFeeReceiver = _marketingFeeReceiver
buyBackReceiver = _buyBackReceiver
pairSwap = _PairSwapLiquidy
pair = _PairLiquidy
```

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	EvocPlus.sol#L7
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 ^0.7.6;
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	EvocPlus.sol#L394,462,799
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
REWARD.transfer(_address, balance)
REWARD.transfer(shareholder, amount)
IBEP20(_tokenAddress).transfer(address(msg.sender),
_tokenAmount)
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	

Roles	Library			
	add	Internal	✓	
	remove	Internal	✓	
	has	Internal		
MinterRole	Implementation			
		Public	✓	-
	isMinter	Public		-
	addMinter	Public	✓	onlyMinter
	removeMinter	Public	✓	onlyMinter
	renounceMinter	Public	✓	-
	_addMinter	Internal	✓	
	_removeMinter	Internal	✓	
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-

	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Auth	Implementation			
		Public	✓	-
	authorize	Public	✓	onlyOwner
	unauthorize	Public	✓	onlyOwner
	isOwner	Public		-
	isAuthorized	Public		-
	setPass	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	renounceOwnership	Public	✓	onlyOwner
IDEXFactory	Interface			
	createPair	External	✓	-
IDEXRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-

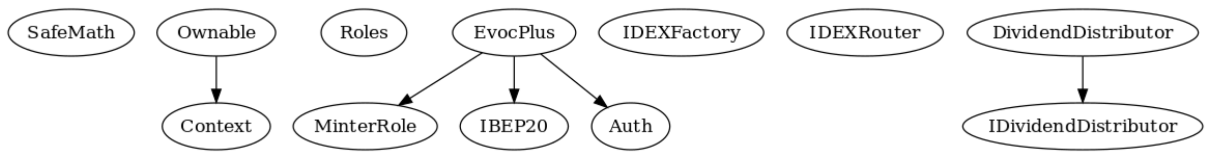
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-
	deposit	External	Payable	-
	process	External	✓	-
DividendDistributor	Implementation	IDividendDistributor		
		Public	✓	-
	setDistributionCriteria	External	✓	onlyToken
	setShare	External	✓	onlyToken
	clearStuckDividends	External	✓	onlyToken
	setRewardToken	External	✓	onlyToken
	setWBNB	External	✓	onlyToken
	deposit	External	Payable	onlyToken
	process	External	✓	onlyToken
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	
	claimDividend	External	✓	-
	getUnpaidEarnings	Public		-

	getCumulativeDividends	Internal		
	addShareholder	Internal	✓	
	removeShareholder	Internal	✓	
EvocPlus	Implementation	IBEP20, Auth, MinterRole		
		Public	✓	Auth
		External	Payable	-
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	Public		-
	allowance	External		-
	maxSupply	External		-
	devListAddress	External	✓	authorized
	approve	Public	✓	-
	approveMax	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	mint	Public	✓	onlyMinter
	_mint	Internal	✓	
	setMaxWalletPercent	External	✓	onlyOwner

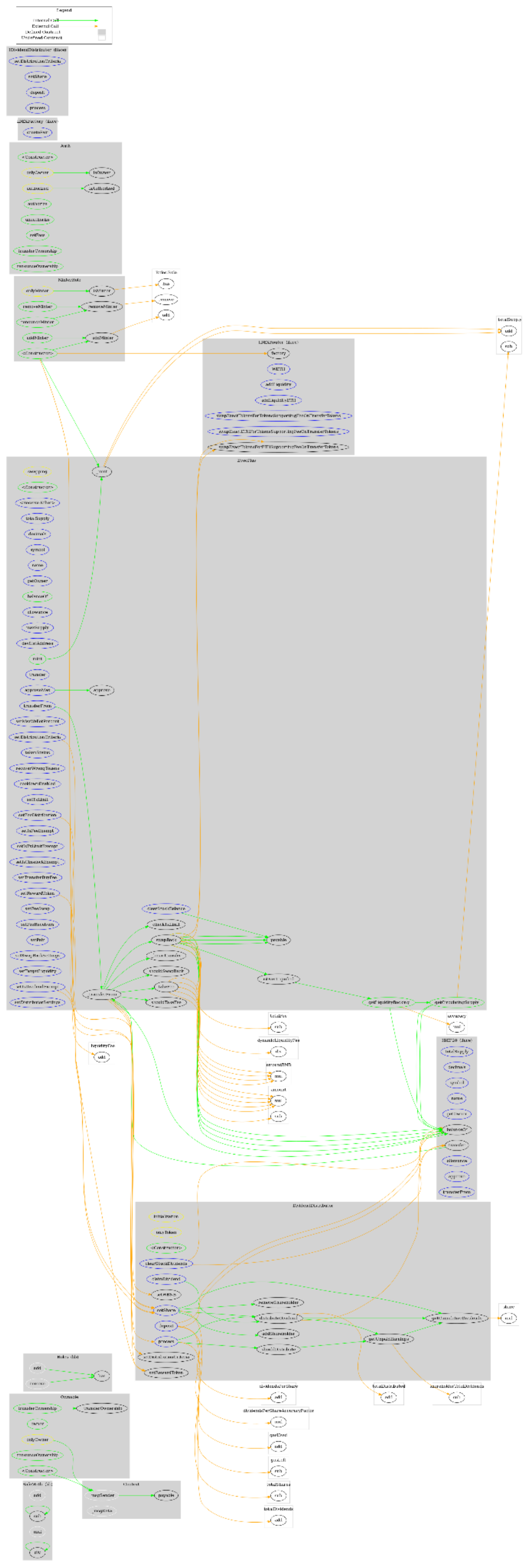
	setRewardToken	External	✓	onlyOwner
	_transferFrom	Internal	✓	
	_basicTransfer	Internal	✓	
	checkTxLimit	Internal		
	shouldTakeFee	Internal		
	takeFee	Internal	✓	
	shouldSwapBack	Internal		
	clearStuckBalance	External	✓	authorized
	tokenStatus	External	✓	onlyOwner
	recoverWrongTokens	External	✓	authorized
	cooldownEnabled	External	✓	onlyOwner
	swapBack	Internal	✓	swapping
	setTxLimit	External	✓	onlyOwner
	setIsDividendExempt	External	✓	onlyOwner
	setIsFeeExempt	External	✓	onlyOwner
	setIsTxLimitExempt	External	✓	onlyOwner
	setIsTimelockExempt	External	✓	onlyOwner
	setTransferBuyFee	External	✓	onlyOwner
	setFeeDistribution	External	✓	onlyOwner
	setFeeSwap	External	✓	onlyOwner
	setFeeReceivers	External	✓	onlyOwner
	setPair	External	✓	onlyOwner
	setSwapBackSettings	External	✓	onlyOwner

	setTargetLiquidity	External	✓	onlyOwner
	setDistributionCriteria	External	✓	onlyOwner
	setDistributorSettings	External	✓	onlyOwner
	getCirculatingSupply	Public		-
	getLiquidityBacking	Public		-
	isOverLiquified	Public		-

Inheritance Graph



Flow Graph



Summary

Evocardano contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. There are some functions that can be abused by the owner like stopping transactions, manipulating the fees, transferring funds to the team's wallet, mint tokens, and blacklist addresses. The contract can be converted into a honeypot and prevent users from selling if the owner abuses the admin functions. if the contract owner abuses the mint functionality, then the contract will be highly inflated. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Team Update

The team responded with the following statement:

"We guarantee that all investors will be informed about the features and their necessity for the project.

We request that these resources be treated as means for our project, according to our justifications. Treated as critics, we may not have a good return on the market because we are a serious project and we guarantee transparency to the entire community."

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