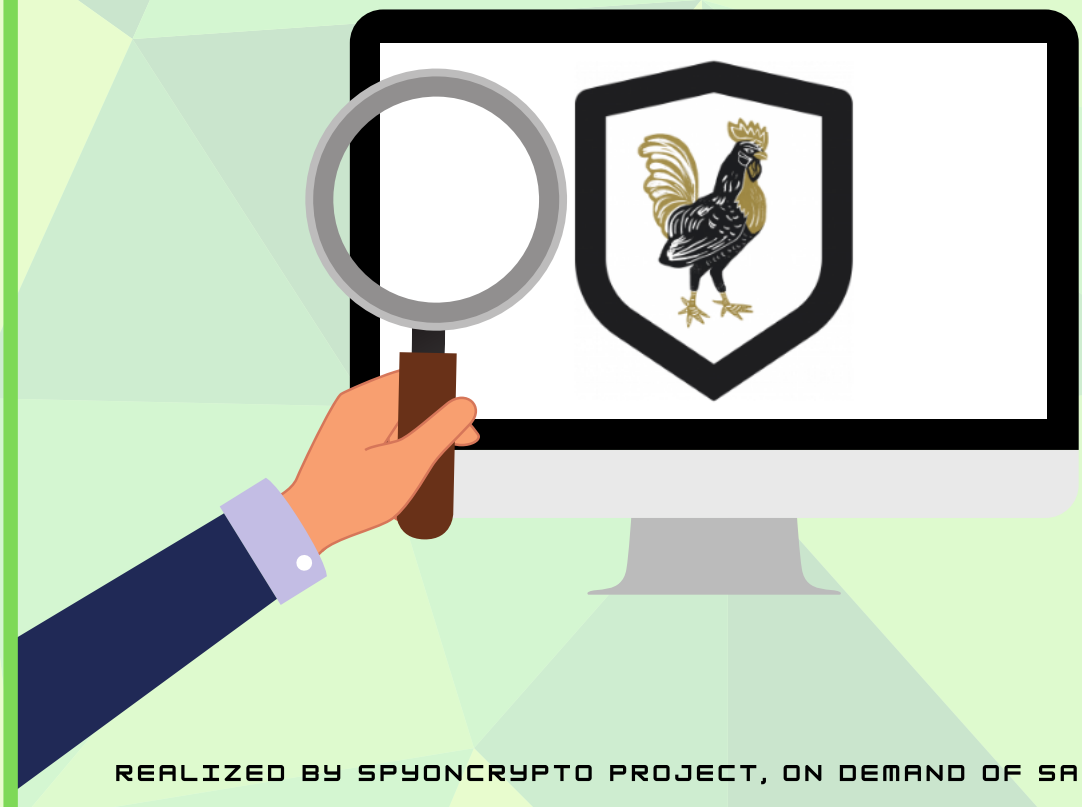
SPY ON CRYPTO

AUDIT SAFECOCK



REALIZED BY SPYONCRYPTO PROJECT, ON DEMAND OF SAFECOCK TEAM

DISCLAIMER

This file is an audit carried out at the request of the interested party.

This report is based on a multitude of analyses and research carried out by our team according to a predefined scheme.

The various steps set out in this file will make it possible to display any vulnerabilities relating to the cybersecurity of the project studied.

These searches are based on the information available to us through the smart contract, but also through information provided by the project developers.

In order to have a better overview of the possible vulnerabilities of this project, the complete reading of this file is recommended.

However, even if this report is available to you, it is only an additional element that can help you in your investigations.

Although a great deal of background work has been done in our investigations, we may have missed some elements, so further research on your part is necessary and advisable.

The conditions mentioned above in the disclaimer are not optional, so if you are not satisfied with them, we strongly urge you to stop reading and analyzing this file and to destroy any copies you have downloaded and/or printed.

These analyses and conclusions are not intended as investment advice. SpyonCrypto is not responsible for any loss of capital, which you are the only owner of.

This report is provided to you as, and without any conditions guaranteed.

SpyonCrypto disclaims any and all liability to the law for any claim or demand by you or any other person for damages.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security.

No product code has been reviewed.

SUMMARY

PRESENTATION

The Safecook is a community project, centred around a deflationary token, part of the fees of which will go directly to a charity.

This project is meant to be "useful" and not just a "meme token", the usefulness of the token will be revealed later by the team.

The project is already launched, there is already a website, various social media and a community on telegram already established.

The different characteristics of the token:

- supply: 69,000,000,000,000 (69trillion)
- Burn wallet: 50% of the total supply
- Allocation team: 1% of the total supply

The different characteristics of social networks:

- Telegram: more than 800 person
- Twitter: more than 500 person
- Reddit: active
- Instagram: yes

CONTRACT DETAILS

CONTRACT NAME
SAFECOCK
SUBMITTED FOR VERIFICATION AT BSCSCAN
2021-06-09

CONTRACT ADDRESS
0X2203BC8D8F238505D8FFCB23BDCE347FB429B168

TOTAL SUPPLY
6,9E+22

TOKEN TICKER
SAFECOCK

DECIMALS
9

TOKEN HOLDERS
2321

TRANSACTIONS COUNT
15569

TOP 100 HOLDERS DOMINANCE
90,04%

CONTRACT DEPLOYER ADDRESS
0X6601E735E8C3110F2FED9CB5E609EAD929E00EE2

CONTRACT'S CURRENT OXNER ADDRESS
0X0000000000000000000000000000000000000000

TOTAL FEES
537745944671994545988O

PANCAKE V2 PAIR
0X4FF5B19E7E16912409567556F3DBC060C303BF15

PANCAKE V2 ROUTER
0X10ED43C718714EB63D5AA57B78B54704E256024E

MAX TRANSACTION AMOUNT
5E+20

DEPLOYED AT TRANSACTION
0XFC1007533201F84E92DDE3807637A354FB96797F8DB8D27EC4D12E2565F
B2BAE

CREATED
2021-05-15 00:35:27 UTC IN BLOCK 7417234

CREATOR
0X6601E735E8C3110F2FED9CB5E609EAD929E00EE2

GRAPHIC ANALYSIS

"SAFECOCK" Token distribution

The top 100 holders collectively own 90.04% (62,130,188,538,057.70 Tokens) of SafeCock

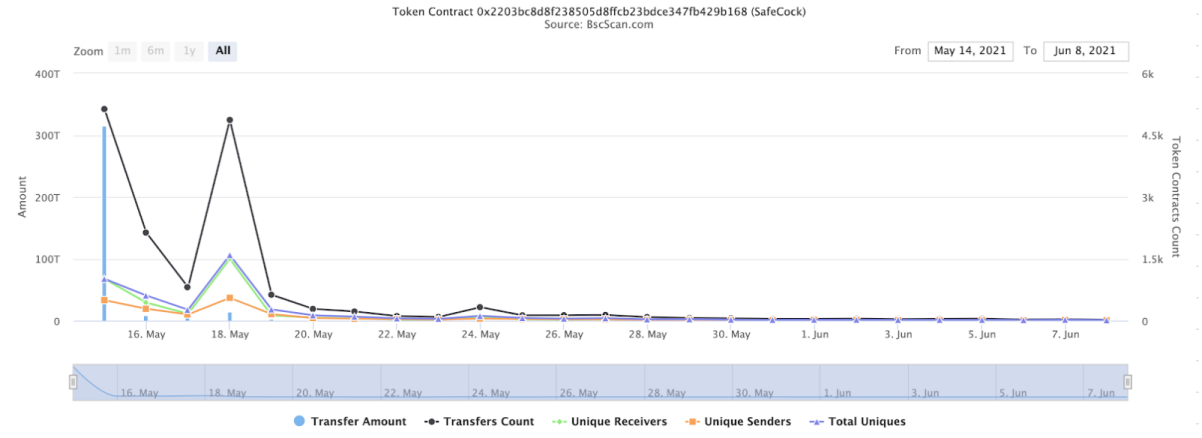
Token Total Supply: 69,000,000,000,000.00 Token | Total Token Holders: 2,321

(A total of 62,130,188,538,057.70 tokens held by the top 100 accounts from the total supply of 69,000,000,000,000.00 token)

"SAFECOCK" contract interaction details

Time Series: Token Contract Overview

Sat 15, May 2021 - Tue 8, Jun 2021



DETECTED VULNERABILITIES

0

28

3

SECURITY ISSUES

MEDIUM

1 - Function could be marked as external.

The function definition of "renounceOwnership" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
236     }
237
238     function renounceOwnership() public virtual onlyOwner {
239         emit OwnershipTransferred(_owner, address(0));
240         _owner = address(0);
241     }
```

```
241     }
242
243     function transferOwnership(address newOwner) public virtual onlyOwner {
244         require(
245             newOwner != address(0),
246             "Ownable: new owner is the zero address"
247         );
248         emit OwnershipTransferred(_owner, newOwner);
249         _owner = newOwner;
250     }
```

The function definition of "geUnlockTime" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
250     }
251
252     function geUnlockTime() public view returns (uint256) {
253         return _lockTime;
254     }
255
```

```
256     //Locks the contract for owner for the amount of time provided
257     function lock(uint256 time) public virtual onlyOwner {
258         _previousOwner = _owner;
259         _owner = address(0);
260         _lockTime = now + time;
261         emit OwnershipTransferred(_owner, address(0));
262     }
```

The function definition of "unlock" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
264     //Unlocks the contract for owner when _lockTime is exceeds
265     function unlock() public virtual {
266         require(
267             _previousOwner == msg.sender,
268             "You don't have permission to unlock"
269         );
270         require(now > _lockTime, "Contract is locked until 7 days");
271         emit OwnershipTransferred(_owner, _previousOwner);
272         _owner = _previousOwner;
273     }
274 }
```

The function definition of "transferDevAddress" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
712     }
713
714     function transferDevAddress(address newDevAddress) public onlyOwner() {
715         devAddress = newDevAddress;
716     }
717
```

The function definition of "name" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
716     }
717
718     function name() public view returns (string memory) {
719         return _name;
720     }
```

The function definition of "symbol" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
720     }
721
722     function symbol() public view returns (string memory) {
723         return _symbol;
724     }
```

The function definition of "decimals" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
724     }
725
726     function decimals() public view returns (uint8) {
727         return _decimals;
728     }
729
```

The function definition of "totalSupply" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
728     }
729
730     function totalSupply() public view override returns (uint256) {
731         return _tTotal;
732     }
733
```

The function definition of "transfer" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
737     }
738
739     function transfer(address recipient, uint256 amount)
740         public
741         override
742         returns (bool)
743     {
744         _transfer(_msgSender(), recipient, amount);
745         return true;
746     }
747
```

The function definition of "allowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
746     }
747
748     function allowance(address owner, address spender)
749         public
750         view
751         override
752         returns (uint256)
753     {
754         return _allowances[owner][spender];
755     }
```

The function definition of "approve" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
756
757     function approve(address spender, uint256 amount)
758         public
759         override
760         returns (bool)
761     {
762         _approve(_msgSender(), spender, amount);
763         return true;
764     }
765
```

The function definition of "transferFrom" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
764     }
765
766     function transferFrom(
767         address sender,
768         address recipient,
769         uint256 amount
770     ) public override returns (bool) {
771         _transfer(sender, recipient, amount);
772         _approve(
773             sender,
774             _msgSender(),
775             _allowances[sender][_msgSender()].sub(
776                 amount,
777                 "ERC20: transfer amount exceeds allowance"
778             )
779         );
780         return true;
781     }
782
```

The function definition of "increaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
781     }
782
783     function increaseAllowance(address spender, uint256 addedValue)
784         public
785         virtual
786         returns (bool)
787     {
788         _approve(
789             _msgSender(),
790             spender,
791             _allowances[_msgSender()][spender].add(addedValue)
792         );
793         return true;
794     }
```

The function definition of "decreaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
794     }
795
796     function decreaseAllowance(address spender, uint256 subtractedValue)
797         public
798         virtual
799         returns (bool)
800     {
801         _approve(
802             _msgSender(),
803             spender,
804             _allowances[_msgSender()][spender].sub(
805                 subtractedValue,
806                 "ERC20: decreased allowance below zero"
807             )
808         );
809         return true;
810     }
811
```

The function definition of "isExcludedFromReward" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
810     }
811
812     function isExcludedFromReward(address account) public view returns (bool) {
813         return _isExcluded[account];
814     }
815
```

The function definition of "totalFees" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
814     }
815
816     function totalFees() public view returns (uint256) {
817         return _tFeeTotal;
818     }
819
```

The function definition of "deliver" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
819
820     function deliver(uint256 tAmount) public {
821         address sender = _msgSender();
822         require(
823             !_isExcluded[sender],
824             "Excluded addresses cannot call this function"
825         );
826         (uint256 rAmount, , , , , ) = _getValues(tAmount);
827         _rOwned[sender] = _rOwned[sender].sub(rAmount);
828         _rTotal = _rTotal.sub(rAmount);
829         _tFeeTotal = _tFeeTotal.add(tAmount);
830     }
831
```

The function definition of "reflectionFromToken" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
830     }
831
832     function reflectionFromToken(uint256 tAmount, bool deductTransferFee)
833         public
834         view
835         returns (uint256)
836     {
837         require(tAmount <= _tTotal, "Amount must be less than supply");
838         if (!deductTransferFee) {
839             (uint256 rAmount, , , , , ) = _getValues(tAmount);
840             return rAmount;
841         } else {
842             (, uint256 rTransferAmount, , , , ) = _getValues(tAmount);
843             return rTransferAmount;
844         }
845     }
846
```

The function definition of "excludeFromReward" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
858     }
859
860     function excludeFromReward(address account) public onlyOwner() {
861         // require(account != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not exclude
    router.');
862         require(!_isExcluded[account], "Account is already excluded");
863         if (_rOwned[account] > 0) {
864             _tOwned[account] = tokenFromReflection(_rOwned[account]);
865         }
866         _isExcluded[account] = true;
867         _excluded.push(account);
868     }
869
```

The function definition of "excludeFromFee" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
903     }
904
905     function excludeFromFee(address account) public onlyOwner {
906         _isExcludedFromFee[account] = true;
907     }
908
```

The function definition of "includeInFee" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
907     }
908
909     function includeInFee(address account) public onlyOwner {
910         _isExcludedFromFee[account] = false;
911     }
912
```

The function definition of "setSwapAndLiquifyEnabled" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants.Consider to mark it as "external" instead.

```
923     }
924
925     function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
926         swapAndLiquifyEnabled = _enabled;
927         emit SwapAndLiquifyEnabledUpdated(_enabled);
928     }
929
```

The function definition of "isExcludedFromFee" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

```
1052    }
1053
1054    function isExcludedFromFee(address account) public view returns (bool) {
1055        return _isExcludedFromFee[account];
1056    }
1057
```

2 - Read of persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

```
683         pancakeV2Router = _pancakeV2Router;
684         //exclude owner and this contract from fee
685         _isExcludedFromFee[owner()] = true;
686         _isExcludedFromFee[devAddress] = true;
687         _isExcludedFromFee[address(this)] = true;
```

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the

call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

```
684         //exclude owner and this contract from fee
685         _isExcludedFromFee[owner()] = true;
686         _isExcludedFromFee[devAddress] = true;
687         _isExcludedFromFee[address(this)] = true;
688         emit Transfer(address(0), _msgSender(), _tTotal);
```

3 - Write to persistent state following external call.

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

```
685         _isExcludedFromFee[owner()] = true;
686         _isExcludedFromFee[devAddress] = true;
687         _isExcludedFromFee[address(this)] = true;
688         emit Transfer(address(0), _msgSender(), _tTotal);
689     }
690
```

LOW

1 - A floating pragma is set.

The current pragma Solidity directive is ""^0.6.12"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

```
5  //SAFECOCK BABY
6  // HOLD TIGHT... BUT NOT TOO TIGHT
7  // SPDX-License-Identifier: Unlicensed
8  pragma solidity ^0.6.12;
9
```

2 - A call to a user-supplied address is executed.

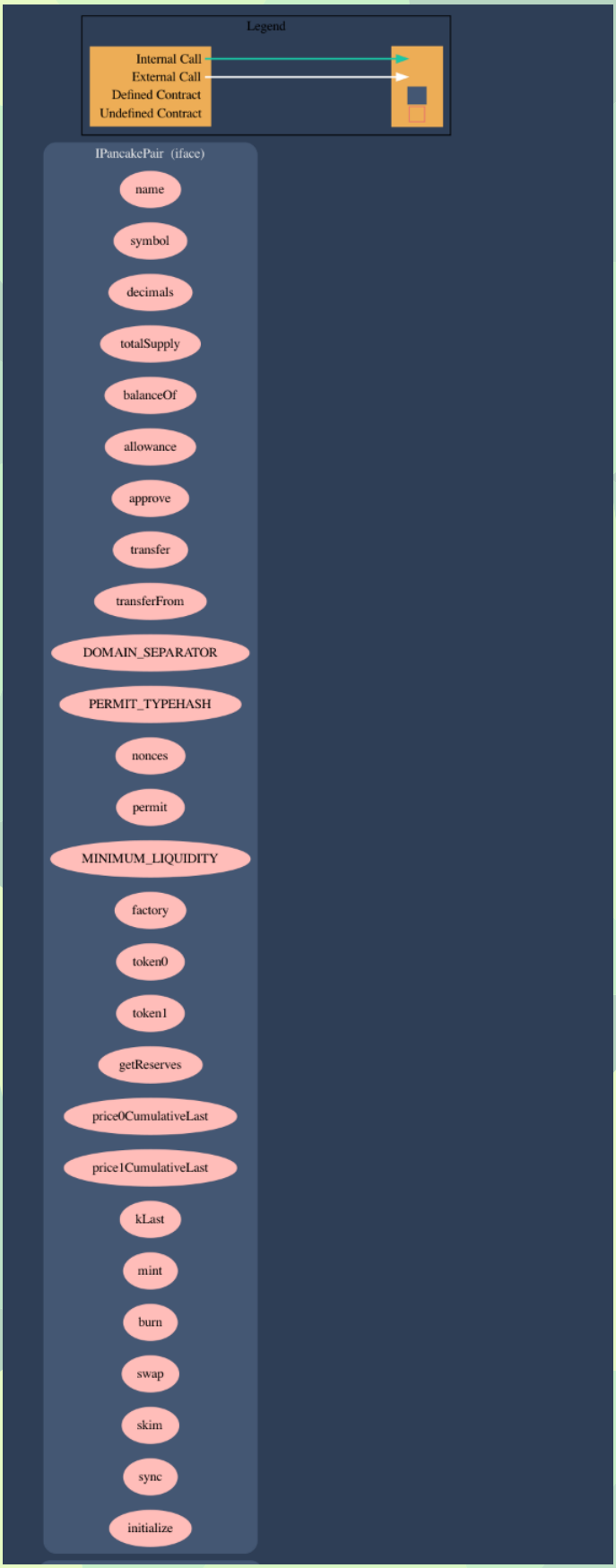
An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this contract. Reentering the contract in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are executed after this call and/or reentrancy guards are in place.

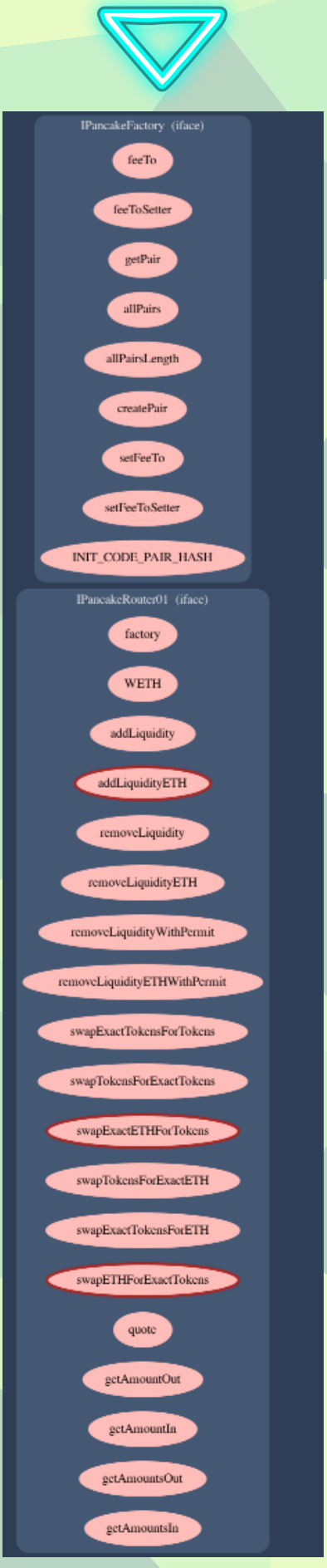
```
677         IPancakeRouter02 _pancakeV2Router =
678             IPancakeRouter02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
679         pancakeV2Pair = IPancakeFactory(_pancakeV2Router.factory()).createPair(
680             address(this),
681             _pancakeV2Router.WETH()
682         );
683         pancakeV2Router = _pancakeV2Router;
684         //exclude owner and this contract from fee
```

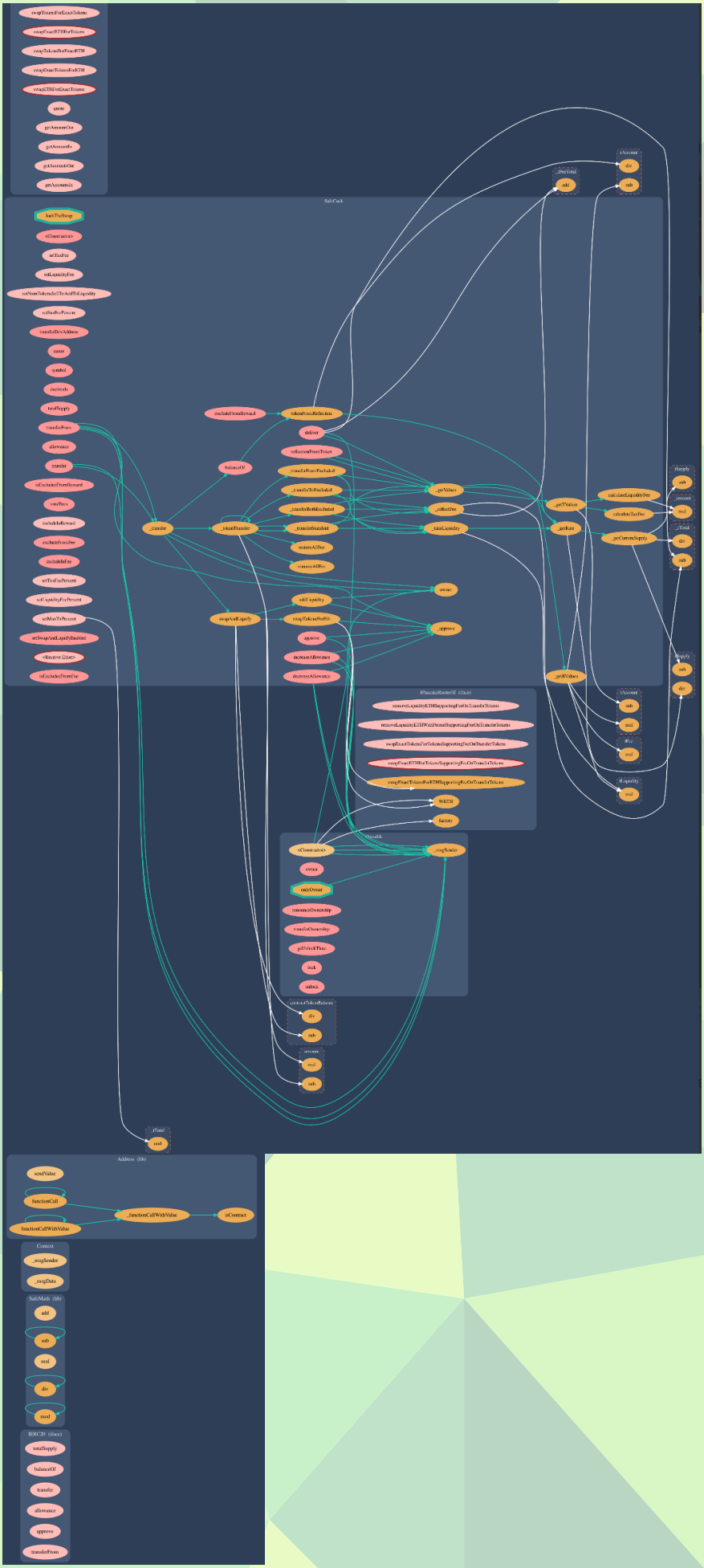
3 - Multiple calls are executed in the same transaction.

This call is executed following another call within the same transaction. It is possible that the call never gets executed if a prior call fails permanently. This might be caused intentionally by a malicious callee. If possible, refactor the code such that each transaction only executes one external call or make sure that all callees can be trusted (i.e. they're part of your own codebase).

```
679         pancakeV2Pair = IPancakeFactory(_pancakeV2Router.factory()).createPair(
680             address(this),
681             _pancakeV2Router.WETH()
682         );
683         pancakeV2Router = _pancakeV2Router;
```


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IPancakePair (iface)
name
symbol
decimals
totalSupply
balanceOf
allowance
approve
transfer
transferFrom
DOMAIN_SEPARATOR
PERMIT_TYPEHASH
nonces
permit
MINIMUM_LIQUIDITY
factory
token0
token1
getReserves
price0CumulativeLast
price1CumulativeLast
kLast
mint
burn
swap
skim
sync
initialize


IPancakeFactory (iface)
feeTo
feeToSetter
getPair
allPairs
allPairsLength
createPair
setFeeTo
setFeeToSetter
INIT_CODE_PAIR_HASH
IPancakeRouter01 (iface)
factory
WETH
addLiquidity
addLiquidityETH
removeLiquidity
removeLiquidityETH
removeLiquidityWithPermit
removeLiquidityETHWithPermit
swapExactTokensForTokens
swapTokensForExactTokens
swapExactETHForTokens
swapTokensForExactETH
swapExactTokensForETH
swapETHForExactTokens
quote
getAmountOut
getAmountIn
getAmountsOut
getAmountsIn


swapTokensForExactTokens
swapTokensForExactETH
swapExactTokensForETH
swapETHForExactTokens
quote
getAmountOut
getAmountIn
getAmountsOut
getAmountsIn
SafeCock
_transfer
_tokenTransfer
_takeLiquidity
_getValues
_getRate
_getTValues
_getRValues
_getCurrentSupply
_reflectFee
_transferStandard
_transferToExcluded
_transferFromExcluded
_transferBothExcluded
tokenFromReflection
deliver
reflectionFromToken
excludeFromReward
includeInReward
excludeFromFee
includeInFee
setTaxFeePercent
setLiquidityFeePercent
setMaxTxPercent
setSwapAndLiquifyEnabled
isExcludedFromFee
totalFees
sendETHToFee
swapAndLiquify
swapTokensForEth
addLiquidity
approve
increaseAllowance
decreaseAllowance
balanceOf
transferFrom
allowance
transfer
isExcludedFromReward
totalSupply
name
symbol
decimals
minTokensSellToAddToLiquidity
swapAndLiquifyEnabled
<Receive Ether>
<Constructor>
owner
onlyOwner
renounceOwnership
transferOwnership
getUnlockTime
lock
unlock
_msgSender
_msgData
IPancakeRouter02
IPancakeFactory
WETH
factory
IUniswapV2Pair
IERC20
Address (lib)
sendValue
functionCall
functionCallWithValue
isContract
Context
SafeMath (lib)
add
sub
mul
div
mod
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom

Parent	Function Name	Visibility	Mutability	Modifiers
IERC20	////			
	totalSupply	External		NO
	balanceOf	External		NO
	transfer	External		NO
	allowance	External		NO
	approve	External		NO
	transferFrom	External		NO
SafeMath	////	////	////	////
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
Context	////	////	////	////
	_msgSender	Internal		
	_msgData	Internal		
Address	////	////	////	////
	isContract	Internal		
	sendValue	Internal		
	functionCall	Internal		
	functionCallWithValue	Internal		
	_functionCallWithValue	Private		
Ownable	////	////	////	////
	<Constructor>			
	owner	Public		NO
	renounceOwnership	Public		onlyOwner
	transferOwnership	Public		onlyOwner
	geUnlockTime	Public		NO
	lock	Public		onlyOwner
	unlock	Public		NO
IPancakeRouter01	////	////	////	////
	factory	External		NO
	WETH	External		NO
	addLiquidity	External		NO
	addLiquidityETH	External		NO
	removeLiquidity	External		NO
	removeLiquidityETH	External		NO
	removeLiquidityWithPermit	External		NO
	removeLiquidityETHWithPermit	External		NO
	swapExactTokensForTokens	External		NO
	swapTokensForExactTokens	External		NO
	swapExactETHForTokens	External		NO
	swapTokensForExactETH	External		NO
	swapExactTokensForETH	External		NO
	swapETHForExactTokens	External		NO
	quote	External		NO
	getAmountOut	External		NO
	getAmountIn	External		NO
	getAmountsOut	External		NO
	getAmountsIn	External		NO
IPancakeRouter02	////	////	////	////
	removeLiquidityETHSupportingFeeOnTransferTokens	External		NO
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External		NO
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External		NO
	swapExactETHForTokensSupportingFeeOnTransferTokens	External		NO
	swapExactTokensForETHSupportingFeeOnTransferTokens	External		NO
IPancakeFactory	////	////	////	////
	feeTo	External		NO
	feeToSetter	External		NO
	getPair	External		NO
	allPairs	External		NO
	allPairsLength	External		NO
	createPair	External		NO
	setFeeTo	External		NO
	setFeeToSetter	External		NO
	INIT_CODE_PAIR_HASH	External		NO

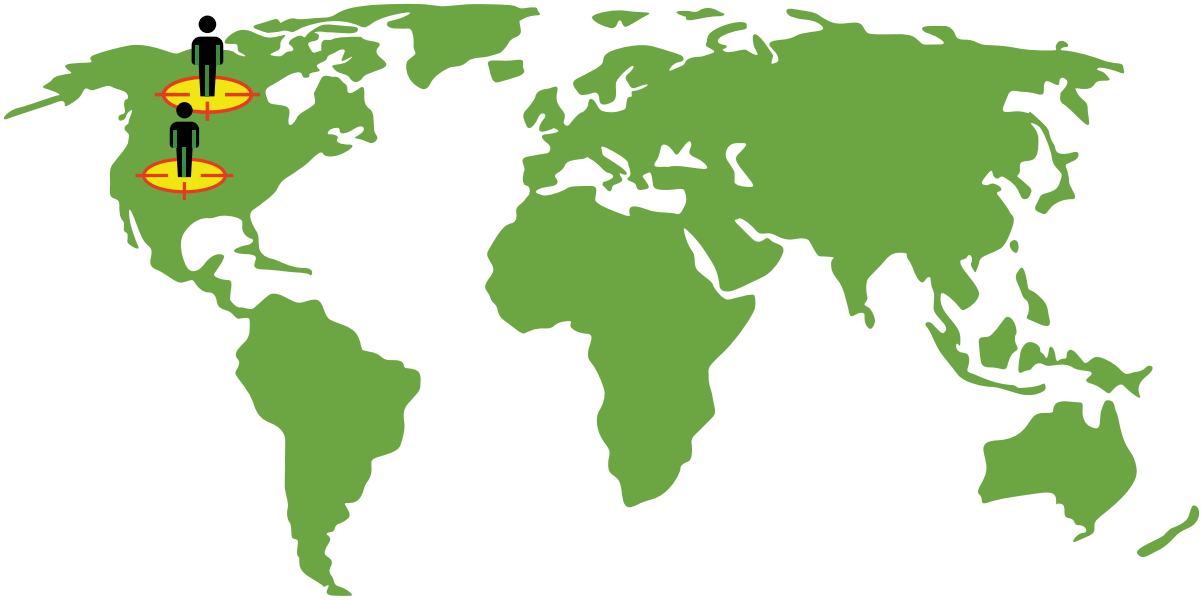
IPancakePair	////	////	////	////
	name	External !		NO!
	symbol	External !		NO!
	decimals	External !		NO!
	totalSupply	External !		NO!
	balanceOf	External !		NO!
	allowance	External !		NO!
	approve	External !	🛑	NO!
	transfer	External !	🛑	NO!
	transferFrom	External !	🛑	NO!
	DOMAIN_SEPARATOR	External !		NO!
	PERMIT_TYPEHASH	External !		NO!
	nonces	External !		NO!
	permit	External !	🛑	NO!
	MINIMUM_LIQUIDITY	External !		NO!
	factory	External !		NO!
	token0	External !		NO!
	token1	External !		NO!
	getReserves	External !		NO!
	price0CumulativeLast	External !		NO!
	price1CumulativeLast	External !		NO!
	kLast	External !		NO!
	mint	External !	🛑	NO!
	burn	External !	🛑	NO!
	swap	External !	🛑	NO!
	skim	External !	🛑	NO!
	sync	External !	🛑	NO!
	initialize	External !	🛑	NO!
SafeCock	////	////	////	////
	<Constructor>	Public !	🛑	NO!
	setTaxFee	External !	🛑	onlyOwner
	setLiquidityFee	External !	🛑	onlyOwner
	setNumTokensSellToAddToLiquidity	External !	🛑	onlyOwner
	setSusFeePercent	External !	🛑	onlyOwner
	transferDevAddress	Public !	🛑	onlyOwner
	name	Public !		NO!
	symbol	Public !		NO!
	decimals	Public !		NO!
	totalSupply	Public !		NO!
	balanceOf	Public !		NO!
	transfer	Public !	🛑	NO!
	allowance	Public !		NO!
	approve	Public !	🛑	NO!
	transferFrom	Public !	🛑	NO!
	increaseAllowance	Public !	🛑	NO!
	decreaseAllowance	Public !	🛑	NO!

	isExcludedFromReward	Public ❗		NO❗
	totalFees	Public ❗		NO❗
	deliver	Public ❗	🔴	NO❗
	reflectionFromToken	Public ❗		NO❗
	tokenFromReflection	Public ❗		NO❗
	excludeFromReward	Public ❗	🔴	onlyOwner
	includeInReward	External ❗	🔴	onlyOwner
	_transferBothExcluded	Private 🔐	🔴	
	excludeFromFee	Public ❗	🔴	onlyOwner
	includeInFee	Public ❗	🔴	onlyOwner
	setTaxFeePercent	External ❗	🔴	onlyOwner
	setLiquidityFeePercent	External ❗	🔴	onlyOwner
	setMaxTxPercent	External ❗	🔴	onlyOwner
	setSwapAndLiquifyEnabled	Public ❗	🔴	onlyOwner
	<Receive Ether>	External ❗		NO❗
	_reflectFee	Private 🔐	🔴	
	_getValues	Private 🔐		
	_getTValues	Private 🔐		
	_getRValues	Private 🔐		
	_getRate	Private 🔐		
	_getCurrentSupply	Private 🔐		
	_takeLiquidity	Private 🔐	🔴	
	calculateTaxFee	Private 🔐		
	calculateLiquidityFee	Private 🔐		
	removeAllFee	Private 🔐	🔴	
	restoreAllFee	Private 🔐	🔴	
	isExcludedFromFee	Public ❗		NO❗
	_approve	Private 🔐	🔴	
	_transfer	Private 🔐	🔴	
	swapAndLiquify	Private 🔐	🔴	lockTheSwap
	swapTokensForEth	Private 🔐	🔴	
	addLiquidity	Private 🔐	🔴	
	_tokenTransfer	Private 🔐	🔴	
	_transferStandard	Private 🔐	🔴	
	_transferToExcluded	Private 🔐	🔴	
	_transferFromExcluded	Private 🔐	🔴	

🔸 = Function can modify state

LOCATION TEAM

TEAM CANADA AND USA



SOCIAL MEDIA

@safecockbsc

Safecockbsc

Safecock

https://vm.tiktok.com/ZMeW9xjyJ/

Safecockbsc

https://safecock.medium.com/

https://discord.gg/zCs3Mw8Xt9

https://youtube.com/channel/UCfFM6i7EqeO32x99rTIwKpQ

https://safecock.finance

NOTE AND CONCLUSION

The smart safecock contract does not reveal any flaws that could be critical for investors and founders.

The ownership of the contract has been renounced, reversed to the owner is blocked, it will be up to the investor and owner to ask and repeat this task over time so that it does not become a problematic risk for investors.

As well as to verify that the sustainability tax of 0,1% is used for promotional purposes, burn, buyback and development of the project

Safecock smart contract and internal mechanism seems safe the safecock economy model does look sustainable for the future of the project and investors on the long run.

TG : SPYONCRYPTO
TG CHAT: SPYONCRYPTO CHAT
TWITTER: @spyoncrypto
MAIL: SOCIAL@SPYONCRYPTO.COM
SITE WEB: WWW.SPYONCRYPTO.COM