Smart Contract Security Audit V1

Succeed Smart Contract Audit

APR 5, 2025

https://saferico.com/

business@saferico.com

_

Table of Contents

Background

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Project Information

- **Platform**: Binance Smart Chain
- **Name**: Succeed
- **Language** : Solidity
- **Contract Address**: 0x3f4e6ce0f4379235a36660c21328c49635beefc8
- **Code Source**: https://bscscan.com/address/0x3f4e6ce0f4379235a36660c21328c49635beefc8#code

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Well-Secured**.

Well Secured	✓
Secured	
Poor Secured	
Insecure	

Automated checks are with remix IDE. All issues were performed by the team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

Team found 0 critical, 0 high, 0 medium, 4 low, 0 very low-level issues and 1 note in all solidity files of the contract

The files:

Succeed.sol

Audit Score:

99% secure

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
Succeed.sol	6c2e02a25ae37de7e4d65412789f4e03e0fe2952	0x3f4e6ce0f4379235a36660c21328c49635beefc8

- Contract: Succeed
- Inherit: ERC20, Ownable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
_isExcludedMaxTransactionAmount	✓	Read / public	**Passed**
automatedMarketMakerPairs	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
allowance	✓	Read / public	**Passed**
buyBurnFee	✓	Read / public	**Passed**
buyDonationFee	✓	Read / public	**Passed**
buyLiquidityFee	✓	Read / public	**Passed**
buyMarketingFee	✓	Read / public	**Passed**
buyTotalFees	✓	Read / public	**Passed**
sellBurnFee	✓	Read / public	**Passed**
sellDonationFee	✓	Read / public	**Passed**
sellLiquidityFee	✓	Read / public	**Passed**
sellMarketingFee	✓	Read / public	**Passed**
sellTotalFees	✓	Read / public	**Passed**

deadAddress	✓	Read / public	**Passed**
decimals	✓	Read / public	**Passed**
donationWallet	✓	Read / public	**Passed**
marketingWallet	✓	Read / public	**Passed**
isExcludedFromFees	✓	Read / public	**Passed**
maxTransactionAmount	✓	Read / public	**Passed**
name	✓	Read / public	**Passed**
owner	✓	Read / public	**Passed**
swapEnabled	✓	Read / public	**Passed**
swapTokensAtAmount	✓	Read / public	**Passed**
symbol	✓	Read / public	**Passed**
tokensForBurn	✓	Read / public	**Passed**
tokensForDonation	✓	Read / public	**Passed**
tokensForLiquidity	✓	Read / public	**Passed**
tokensFormarketing	✓	Read / public	**Passed**
uniswapV2Pair	✓	Read / public	**Passed**
uniswapV2Router	✓	Read / public	**Passed**
approve	✓	Write / public	**Passed**
buyBackTokens	✓	Write / public	**Passed**
decreaseAllowance	✓	Write / public	**Passed**
increaseAllowance	✓	Write / public	**Passed**
setAutomatedMarketMakerPair	✓	Write / public	**Passed**
updatedonationWallet	✓	Write / public	**Passed**
transfer	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
updateSwapTokensAtAmount	✓	Write / public	**Passed**
updateMarketingWallet	✓	Write / public	**Passed**
excludeFromMaxTransaction	✓	Write / public	**Passed**

excludeFromFees	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
updateMaxAmount	✓	Write / public	**Passed**
updateBuyFees	✓	Write / public	**Passed**
updateSellFees	✓	Write / public	**Passed**
updateSwapEnabled	✓	Write / public	**Passed**

Issues Checking Status

SWC Attack Analysis

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) for more info check https://swcregistry.io/

No.	Issue Description	Checking Status
136	Unencrypted Private Data On-Chain	**Passed**
135	Code With No Effects	**Passed**
134	Message call with hardcoded gas amount	**Passed**
133	Hash Collisions With Multiple Variable Length Arguments	**Passed**
132	Unexpected Ether balance	**Passed**
131	Presence of unused variables	**Passed**
130	Right-To-Left-Override control character (U+202E)	**Passed**
129	Typographical Error	**Passed**
128	DoS with block gas limit.	**Passed**
127	Arbitrary Jump with Function Type Variable	**Passed**
126	Insufficient Gas Griefing	**Passed**
125	Incorrect Inheritance Order	**Passed**
124	Write to Arbitrary Storage Location	**Passed**
123	Requirement Violation	**Passed**
122	Lack of Proper Signature Verification	**Passed**
121	Missing Protection against Signature Replay Attacks	**Passed**
120	Weak Sources of Randomness from Chain Attributes	**Passed**
119	Shadowing State Variables	**Passed**

118	Incorrect Constructor Name	**Passed**
117	Signature Malleability	**Passed**
116	Block values as a proxy for time	**Not Passed**
115	Authorization through tx.origin	**Passed**
114	Transaction Order Dependence	**Passed**
113	DoS with Failed Call	**Passed**
112	Delegatecall to Untrusted Callee	**Passed**
111	Use of Deprecated Solidity Functions	**Passed**
110	Assert Violation	**Passed**
109	Uninitialized Storage Pointer	**Passed**
108	State Variable Default Visibility	**Passed**
107	Reentrancy	**Passed**
106	Unprotected SELFDESTRUCT Instruction	**Passed**
105	Unprotected Ether Withdrawal	**Passed**
104	Unchecked Call Return Value	**Passed**
103	Floating Pragma	**Not Passed**
102	Outdated Compiler Version	**Passed**
101	Integer Overflow and Underflow	**Passed**
100	Function Default Visibility	**Passed**

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can’t lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can’t have significant impact on execution
Note	Lowest-level vulnerabilities, code style violations and info statements can’t affect smart contract execution and can be ignored.

Audit Findings

Critical:

No Critical severity vulnerabilities were found.

High:

No High severity vulnerabilities were found.

Medium:

No Medium severity vulnerabilities were found.

Low:

#Owner privileges (In the period when the owner isn't renounced)

Description

The owner can change the buy and sell fees.
The owner excludes From Fees any address.
The owner update Swap Enabled.

```
    function updateSwapEnabled(bool enabled) external onlyOwner(){
        swapEnabled = enabled;
    }

    function updateBuyFees(uint256 _marketingFee, uint256 _liquidityFee, uint256
_donationFee, uint256 _burnFee) external onlyOwner {
        buyMarketingFee = _marketingFee;
        buyLiquidityFee = _liquidityFee;
        buyDonationFee = _donationFee;
        buyBurnFee = _burnFee;
        buyTotalFees = buyMarketingFee + buyLiquidityFee + buyDonationFee +
buyBurnFee;
        require(buyTotalFees <= 15, "Must keep fees at 15% or less");
    }

    function updateSellFees(uint256 _marketingFee, uint256 _liquidityFee, uint256
_donationFee, uint256 _burnFee) external onlyOwner {
        sellMarketingFee = _marketingFee;
        sellLiquidityFee = _liquidityFee;
        sellDonationFee = _donationFee;
        sellBurnFee = _burnFee;
        sellTotalFees = sellMarketingFee + sellLiquidityFee + sellDonationFee +
sellBurnFee;
        require(sellTotalFees <= 15, "Must keep fees at 30% or less");
```

```
    }

    function excludeFromFees(address account, bool excluded) public onlyOwner {
        _isExcludedFromFees[account] = excluded;
        emit ExcludeFromFees(account, excluded);
    }
```

Remediation

Make these functions internal in next version or the team should announce the investors before doing anything to give them time if they want to do anything.

P.S: This issue is common to the majority of those smart contracts.

Status: Acknowledged.

Use of block.timestamp for comparisons

The value of block.timestamp can be manipulated by the miner. And conditions with strict equality is difficult to achieve - block.timestamp.

```
function swapTokensForEth(uint256 tokenAmount) private {

        // generate the uniswap pair path of token -> weth
        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = uniswapV2Router.WETH();

        _approve(address(this), address(uniswapV2Router),
tokenAmount);

        // make the swap

uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
            tokenAmount,
            0, // accept any amount of ETH
            path,
            address(this),
            block.timestamp
        );

    }
```

Recommendation

Avoid use of block.timestamp.

Status

Acknowledged.

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.29 instead of ^0.8.26). contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors. And avoid Solidity compiler Bugs check here

https://sepolia.etherscan.io/solcbuginfo

Remediation

Remove the ^ sign to lock the pragma version.

Status: Acknowledged.

#Missing zero address validation

When the owner wants update marketing and donation wallets addresses, he has to check for the zero address to make. Otherwise, the function will not work fine.

```
function updateMarketingWallet(address newMarketingWallet) external onlyOwner {
        emit marketingWalletUpdated(newMarketingWallet, marketingWallet);
        marketingWallet = newMarketingWallet;
    }

    function updatedonationWallet(address newWallet) external onlyOwner {
        emit donationWalletUpdated(newWallet, donationWallet);
        donationWallet = newWallet;
    }
```

Remediation

Use the require statement to check for zero addresses.

Status: Acknowledged.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#USE SELFBALANCE() INSTEAD OF ADDRESS(THIS).BALANCE

In Solidity, efficient use of gas is paramount to ensure cost-effective execution on the Ethereum blockchain. Gas can be optimized when obtaining contract balance by using selfbalance() rather than address(this).balance because it bypasses gas costs and refunds, which are not required for obtaining the contract's balance.

```
uint256 initialETHBalance = address(this).balance;

        swapTokensForEth(amountToSwapForETH);

        uint256 ethBalance =
address(this).balance.sub(initialETHBalance);
```

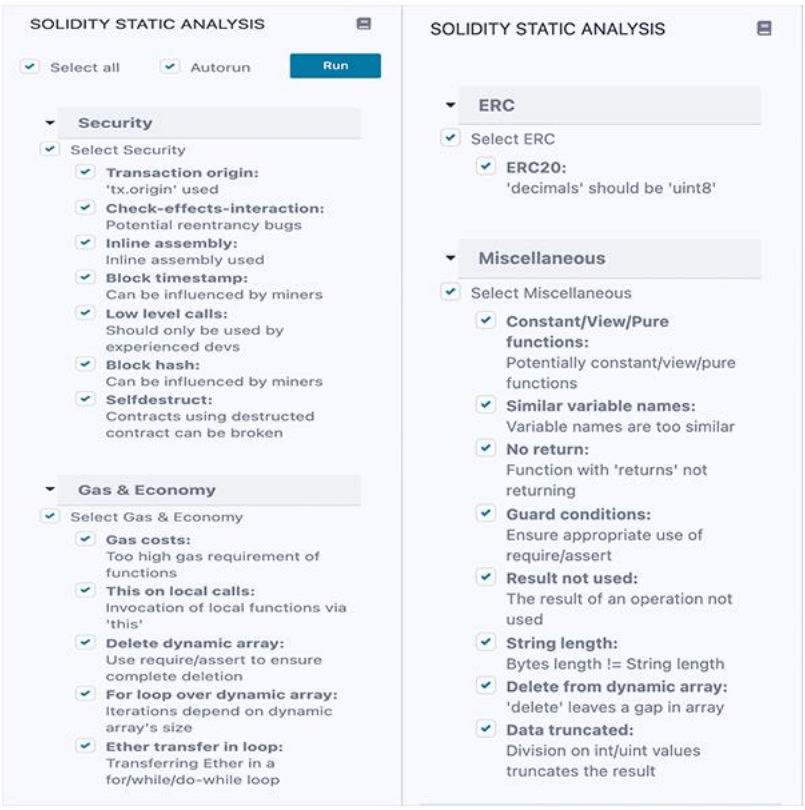
Remediation

To rectify this issue, developers are encouraged to replace instances of address(this).balance with selfbalance() wherever applicable. This optimization not only ensures streamlined gas operations but also contributes to substantial cost savings during contract execution.

Status: Acknowledged.

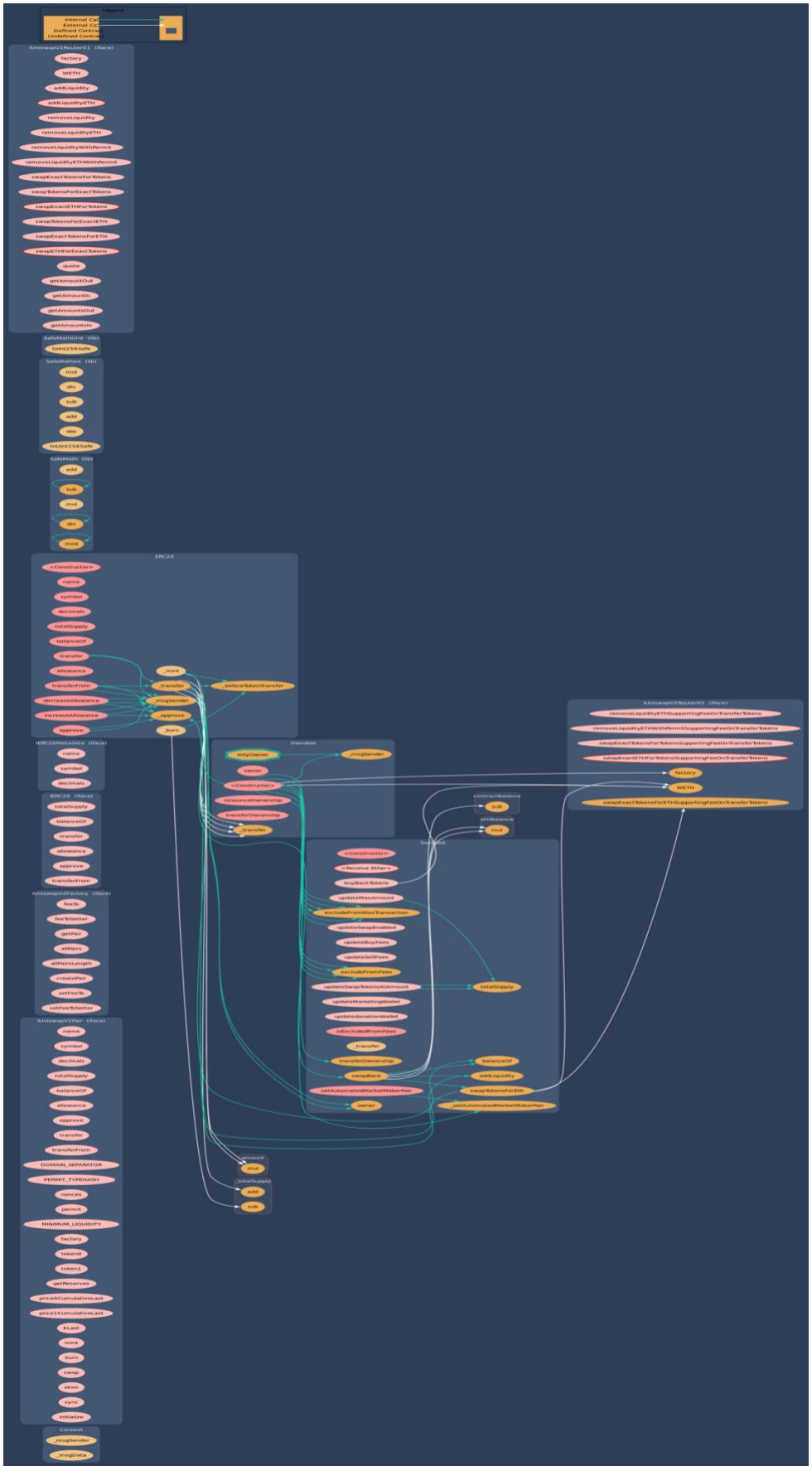
Automatic Testing

1- SOLIDITY STATIC ANALYSIS

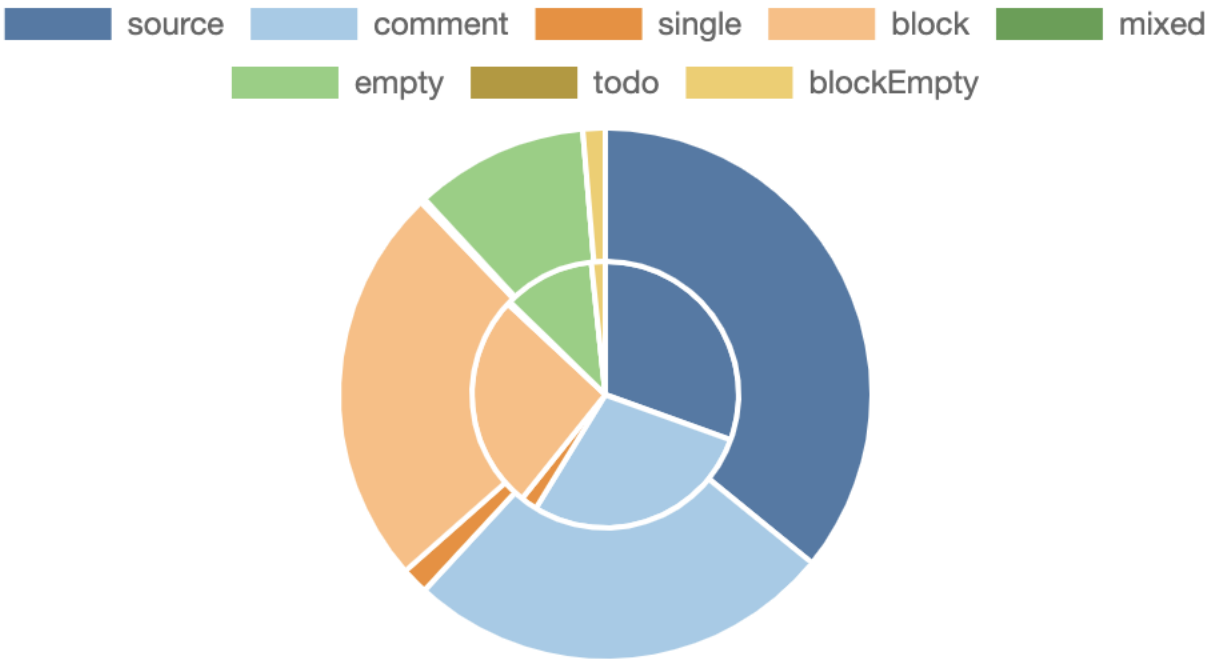


2- Inheritance graph

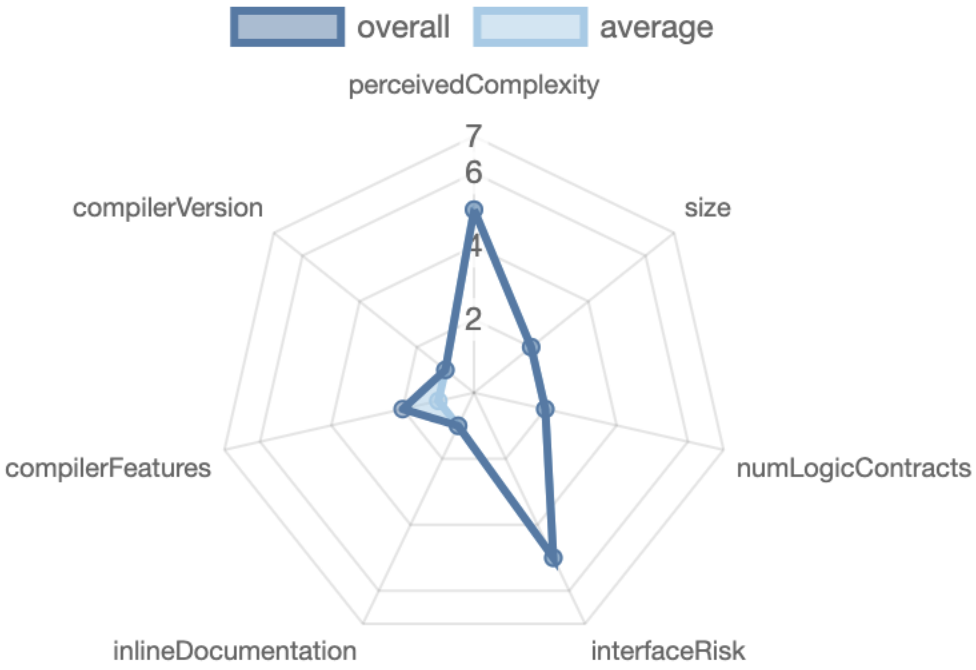
3- Call graph



Source lines



Risk level



Source units in scope

Source Units in Scope

Source Units Analyzed: `1`
Source Units in Scope: `1` **(100%)**

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🎨	Succeed.sol	7	6	1252	970	437	460	502	💰🔆
📝📚🔍🎨	**Totals**	**7**	**6**	**1252**	**970**	**437**	**460**	**502**	💰🔆

Legend: [➖]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Capabilities

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
3	3	6	1

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐Public	💰Payable
95	5

External	Internal	Private	Pure	View
77	83	4	25	35

StateVariables

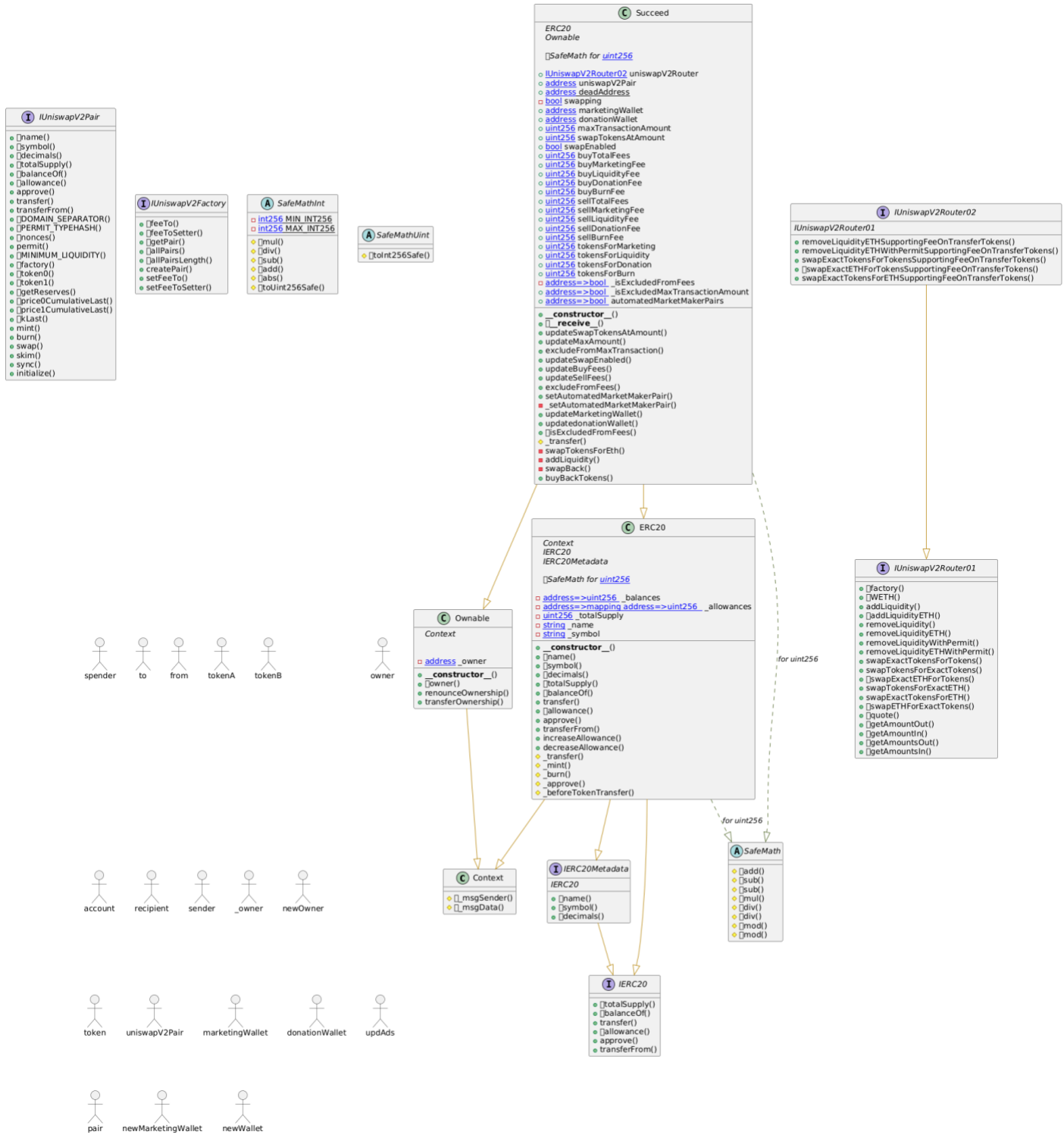
Total	🌐Public
34	24

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
`^0.8.26`		`yes`		

📩 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧱 Uses Hash Functions	🪚 ECRecover	🌀 New/Create/Create2

Unified Modeling Language (UML)



Functions signature

Function Name	Sighash	Function Signature
name	06fdde03	name()
symbol	95d89b41	symbol()
decimals	313ce567	decimals()
totalSupply	18160ddd	totalSupply()
balanceOf	70a08231	balanceOf(address)
allowance	dd62ed3e	allowance(address,address)
approve	095ea7b3	approve(address,uint256)
transfer	a9059cbb	transfer(address,uint256)
transferFrom	23b872dd	transferFrom(address,address,uint256)
DOMAIN_SEPARATOR	3644e515	DOMAIN_SEPARATOR()
PERMIT_TYPEHASH	30adf81f	PERMIT_TYPEHASH()
nonces	7ecebe00	nonces(address)
permit	d505accf	permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
MINIMUM_LIQUIDITY	ba9a7a56	MINIMUM_LIQUIDITY()
factory	c45a0155	factory()
token0	0dfe1681	token0()
token1	d21220a7	token1()
getReserves	0902f1ac	getReserves()
price0CumulativeLast	5909c0d5	price0CumulativeLast()
price1CumulativeLast	5a3d5493	price1CumulativeLast()
kLast	7464fc3d	kLast()
mint	6a627842	mint(address)
burn	89afcb44	burn(address)
swap	022c0d9f	swap(uint256,uint256,address,bytes)
skim	bc25cf77	skim(address)
sync	fff6cae9	sync()
initialize	c4d66de8	initialize(address)
feeTo	017e7e58	feeTo()
feeToSetter	094b7415	feeToSetter()
getPair	e6a43905	getPair(address,address)
allPairs	1e3dd18b	allPairs(uint256)
allPairsLength	574f2ba3	allPairsLength()
createPair	c9c65396	createPair(address,address)
setFeeTo	f46901ed	setFeeTo(address)
setFeeToSetter	a2e74af6	setFeeToSetter(address)
totalSupply	18160ddd	totalSupply()
balanceOf	70a08231	balanceOf(address)
transfer	a9059cbb	transfer(address,uint256)
allowance	dd62ed3e	allowance(address,address)
approve	095ea7b3	approve(address,uint256)
transferFrom	23b872dd	transferFrom(address,address,uint256)
name	06fdde03	name()
symbol	95d89b41	symbol()
decimals	313ce567	decimals()
name	06fdde03	name()
symbol	95d89b41	symbol()
decimals	313ce567	decimals()
totalSupply	18160ddd	totalSupply()
balanceOf	70a08231	balanceOf(address)
transfer	a9059cbb	transfer(address,uint256)
allowance	dd62ed3e	allowance(address,address)
approve	095ea7b3	approve(address,uint256)
transferFrom	23b872dd	transferFrom(address,address,uint256)

increaseAllowance	39509351	increaseAllowance(address,uint256)
decreaseAllowance	a457c2d7	decreaseAllowance(address,uint256)
owner	8da5cb5b	owner()
renounceOwnership	715018a6	renounceOwnership()
transferOwnership	f2fde38b	transferOwnership(address)
factory	c45a0155	factory()
WETH	ad5c4648	WETH()
addLiquidity	e8e33700
addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
addLiquidityETH	f305d719
addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
removeLiquidity	baa2abde
removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
removeLiquidityETH	02751cec
removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
removeLiquidityWithPermit	2195995c
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,b
ool,uint8,bytes32,bytes32)
removeLiquidityETHWithPermit	ded9382a
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,u
int8,bytes32,bytes32)
swapExactTokensForTokens	38ed1739
swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
swapTokensForExactTokens	8803dbee
swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
swapExactETHForTokens	7ff36ab5
swapExactETHForTokens(uint256,address[],address,uint256)
swapTokensForExactETH	4a25d94a
swapTokensForExactETH(uint256,uint256,address[],address,uint256)
swapExactTokensForETH	18cbafe5
swapExactTokensForETH(uint256,uint256,address[],address,uint256)
swapETHForExactTokens	fb3bdb41
swapETHForExactTokens(uint256,address[],address,uint256)
quote	ad615dec	quote(uint256,uint256,uint256)
getAmountOut	054d50d4	getAmountOut(uint256,uint256,uint256)
getAmountIn	85f8c259	getAmountIn(uint256,uint256,uint256)
getAmountsOut	d06ca61f	getAmountsOut(uint256,address[])
getAmountsIn	1f00ca74	getAmountsIn(uint256,address[])
removeLiquidityETHSupportingFeeOnTransferTokens	af2979eb
removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,add
ress,uint256)
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	5b0d5984
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,u
int256,address,uint256,bool,uint8,bytes32,bytes32)
swapExactTokensForTokensSupportingFeeOnTransferTokens	5c11d795
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],add
ress,uint256)
swapExactETHForTokensSupportingFeeOnTransferTokens	b6f9de95
swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint25
6)
swapExactTokensForETHSupportingFeeOnTransferTokens	791ac947
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],addres
s,uint256)
updateSwapTokensAtAmount	d257b34f	updateSwapTokensAtAmount(uint256)
updateMaxAmount	106b5da1	updateMaxAmount(uint256)
excludeFromMaxTransaction	7571336a	excludeFromMaxTransaction(address,bool)
updateSwapEnabled	924de9b7	updateSwapEnabled(bool)
updateBuyFees	2e6ed7ef	updateBuyFees(uint256,uint256,uint256,uint256)
updateSellFees	e7ad9fcd	updateSellFees(uint256,uint256,uint256,uint256)

| excludeFromFees | c0246668 | excludeFromFees(address,bool) |
| setAutomatedMarketMakerPair | 9a7a23d6 |
setAutomatedMarketMakerPair(address,bool) |
updateMarketingWallet	aacebbe3	updateMarketingWallet(address)
updatedonationWallet	23381704	updatedonationWallet(address)
isExcludedFromFees	4fbee193	isExcludedFromFees(address)
buyBackTokens	fc155d1d	buyBackTokens(uint256)

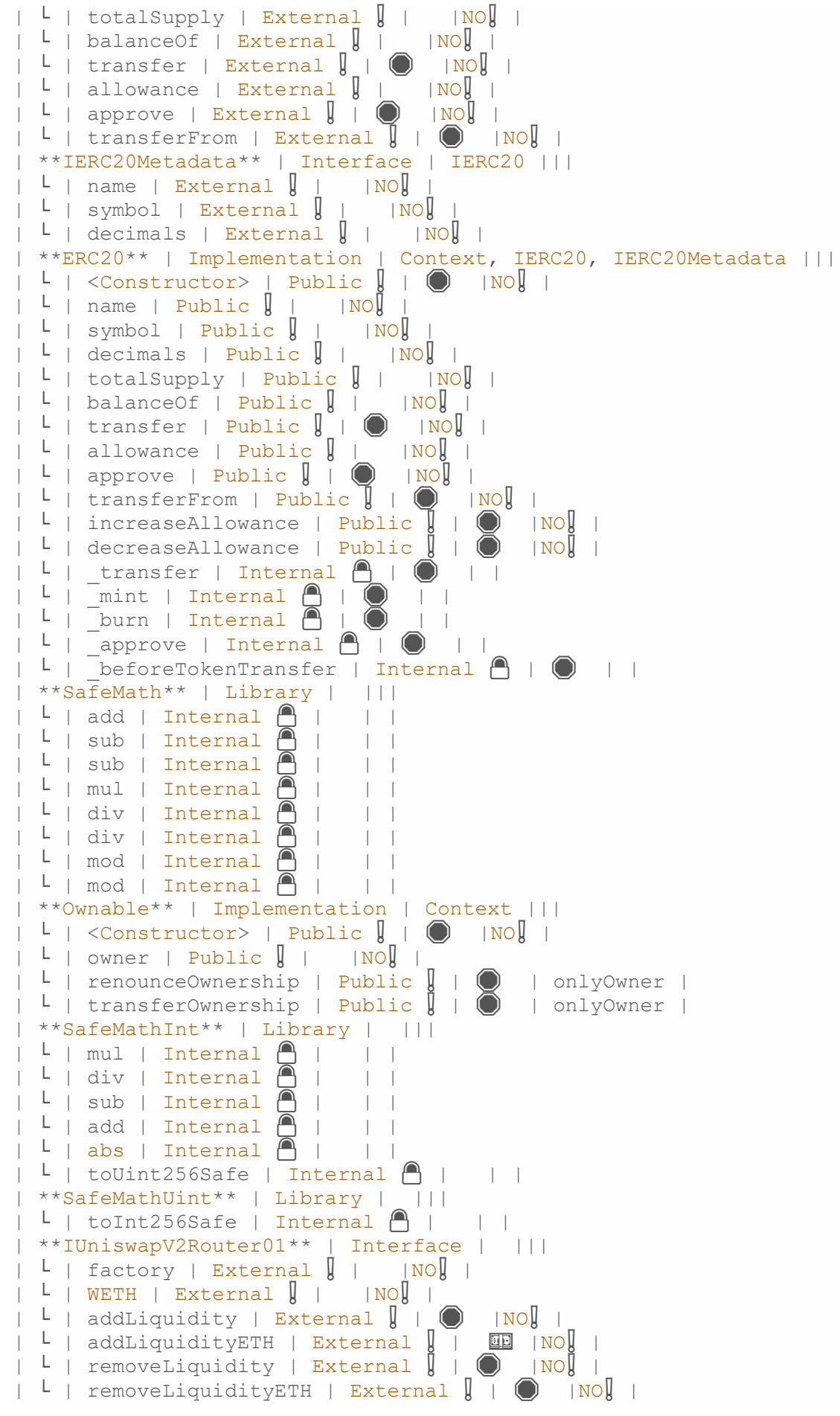
Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/Succeed.sol	6c2e02a25ae37de7e4d65412789f4e03e0fe2952

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IUniswapV2Pair	Interface			
└	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❗
IUniswapV2Factory	Interface			
└	feeTo	External ❗		NO❗
└	feeToSetter	External ❗		NO❗
└	getPair	External ❗		NO❗
└	allPairs	External ❗		NO❗
└	allPairsLength	External ❗		NO❗
└	createPair	External ❗	🛑	NO❗
└	setFeeTo	External ❗	🛑	NO❗
└	setFeeToSetter	External ❗	🛑	NO❗
IERC20	Interface			


└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<Constructor>	Public ❗	🛑	NO❗
└	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❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
SafeMath	Library				
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Ownable	Implementation	Context			
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
SafeMathInt	Library				
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		
└	abs	Internal 🔒		
└	toUint256Safe	Internal 🔒		
SafeMathUint	Library				
└	toInt256Safe	Internal 🔒		
IUniswapV2Router01	Interface				
└	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❗
IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑
NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
Succeed	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	<Receive Ether>	External ❗	💵	NO❗
└	updateSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	updateMaxAmount	External ❗	🛑	onlyOwner
└	excludeFromMaxTransaction	Public ❗	🛑	onlyOwner
└	updateSwapEnabled	External ❗	🛑	onlyOwner
└	updateBuyFees	External ❗	🛑	onlyOwner
└	updateSellFees	External ❗	🛑	onlyOwner
└	excludeFromFees	Public ❗	🛑	onlyOwner
└	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
└	_setAutomatedMarketMakerPair	Private 🔐	🛑	
└	updateMarketingWallet	External ❗	🛑	onlyOwner
└	updatedonationWallet	External ❗	🛑	onlyOwner
└	isExcludedFromFees	Public ❗		NO❗
└	_transfer	Internal 🔒	🛑	
└	swapTokensForEth	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	swapBack	Private 🔐	🛑	
└	buyBackTokens	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Conclusion

The contracts are written systematically. Team found no critical issues. So, it is good to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan Everything.

Security state of the reviewed contract is “Well Secured”.

✓ No volatile code.
✓ No high severity issues were found.

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against the team on the basis of what it says or doesn't say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. team go into more detail on this in the below disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Saferico and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Saferico s) owe no duty of care towards you or any other person, nor does Saferico make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Saferico hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Saferico hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saferico, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. 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.