RugFreeCoins Audit

SAFUPAD Token

Smart Contract Security Audit

July 04, 2021

Contents

Audit details

Audited project

SAFUPAD Token

Contract Address

0xEb4f4d09CcBB3be5bf83c26a42282652913aDB59

Client contact

SAFUPAD Team

Blockchain

Binance smart chain

Project website

https://safupad.io/

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at 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 us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: 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 Rugfreecoins and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Rugfreecoins) owe no duty of care towards you or any other person, nor does Rugfreecoins 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 Rugfreecoins 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, Rugfreecoins hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Rugfreecoins, 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.

Background

Rugfreecoins was commissioned by SAFUPAD Token to perform an audit of the smart contract.

https://bscscan.com/token/0xEb4f4d09CcBB3be5bf83c26a42282652913aDB59

The focus of this audit is to verify that the smart contract is secure, resilient and working according to the specifications.

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

About the project

SAFUPAD token is with a new concept of distributing BNB rewards to the holders while increasing both liquidity and value.

SAFUPAD LAUNCHPAD and SAFUPAD LOCKER are token's upcoming platforms that will bring a huge value to the investors and projects.

- **SAFUPAD LAUNCHPAD**

It is to be introduced for the upcoming projects, and give members a unique opportunity to be involved before launch and listing.

- **SAFUPAD LOCKER**

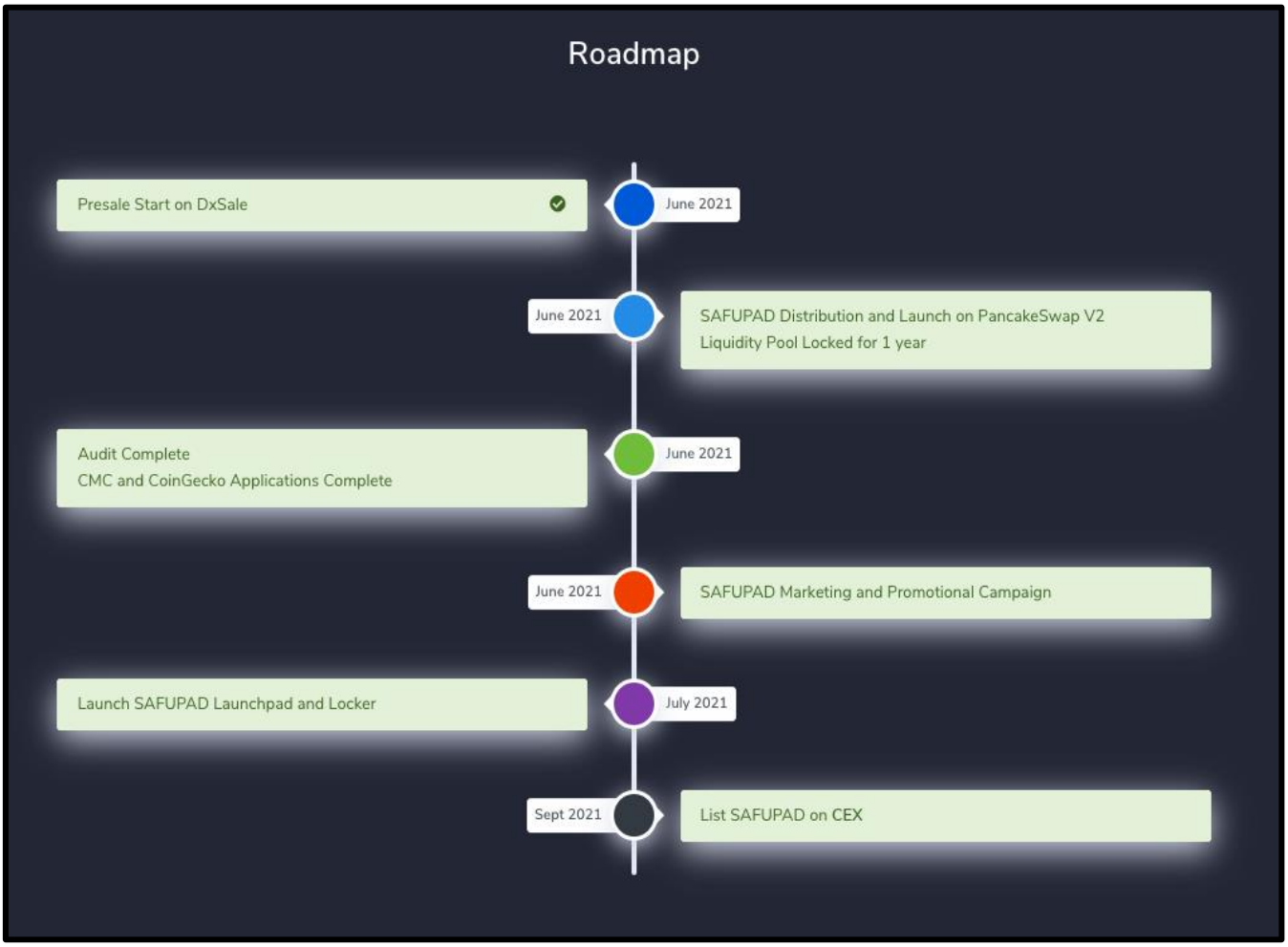
Any project launched on SAFUPAD Launchpad has the ability to lock liquidity through SAFUPAD smart contracts, taking away the ability for those projects to remove or pull liquidity until the end of the set locked period.

Tokenomics

10% fee when buying & selling

- 4% of every trade goes to holders pockets in BNB.
- 4% of every trade goes to the liquidity pool.
- 2% of every trade goes to holders pockets in tokens.

Roadmap


Roadmap
Presale Start on DxSale
June 2021
June 2021
SAFUPAD Distribution and Launch on PancakeSwap V2
Liquidity Pool Locked for 1 year
Audit Complete
CMC and CoinGecko Applications Complete
June 2021
June 2021
SAFUPAD Marketing and Promotional Campaign
Launch SAFUPAD Launchpad and Locker
July 2021
Sept 2021
List SAFUPAD on CEX

Target market and the concept

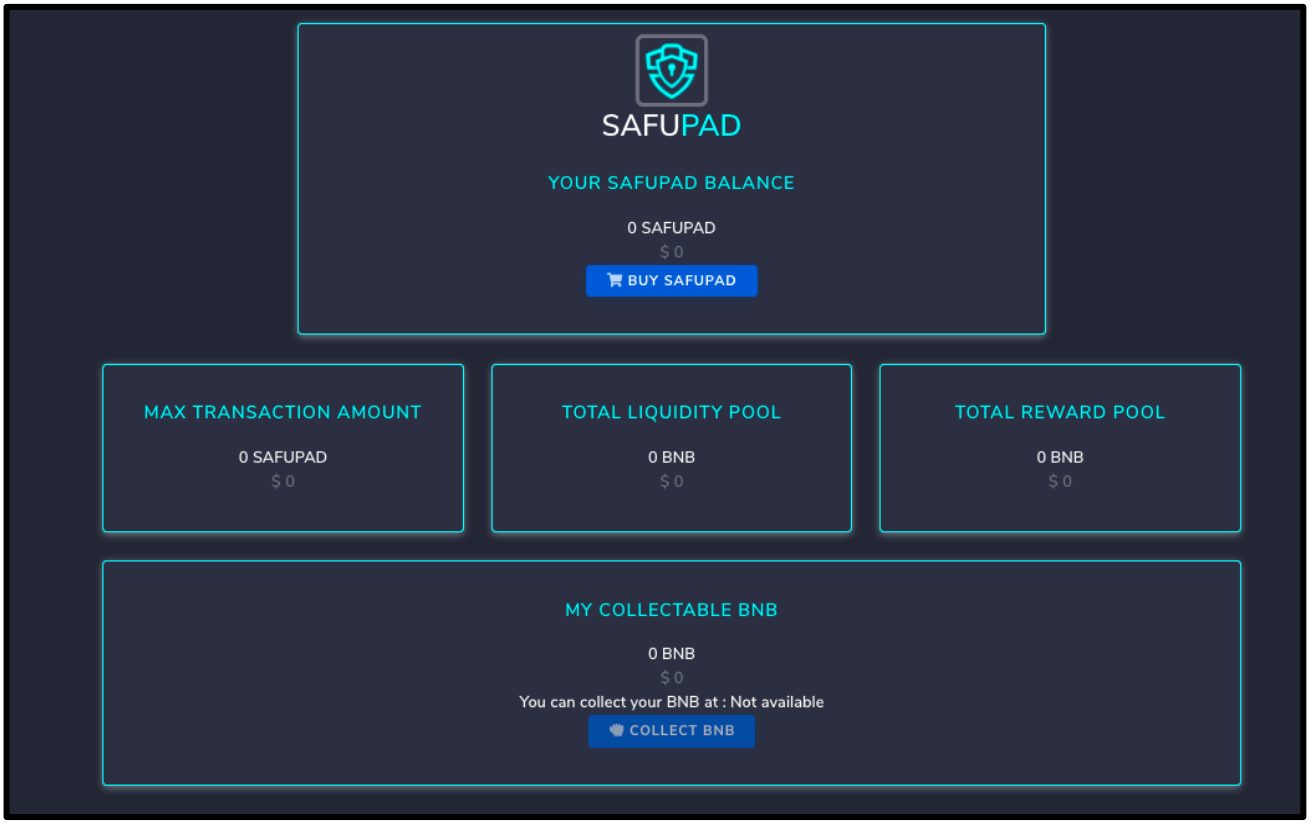
Target market

- Anyone who's interested in Crypto space with long term investment plans.
- Anyone who's ready to earn a passive income by holding tokens.
- Anyone who's interested in trading tokens.
- Anyone who's interested in doubling their BNB rewards through the BNB rewards.
- Anyone who's interested in finding new projects through SAFUPAD upcoming platform.
- Any project to lock their liquidity through a certain period through SAFUPAD upcoming platform.

Core concept

➢ **BNB rewards**

A 4% of tokens will be deducted as a fee and swapped into BNB and will be sent to the BNB pool. Once a week SAFUPAD holders can claim a percentage of it based on their holding share against the supply. Users can claim their BNB through the DAPP. This mechanism encourages holding instead of selling.



- **BNB reward lottery**

When holders are claiming their BNB rewards, random numbers are generated around 1 to 100. If the generated random number is less than 5, the holder's rewards will be multiplied by 1.5 or 2.

- **Coin Burning mechanism**

30% of tokens have been burned in the launch. 2% of tokens go to the burn wallet in each transaction in the distribution, which is making burn wallet-size get increased, which leads to a decrease in the total supply from circulation. Prices also will go higher when the supply is lower.

When claiming BNBs, if the BNB amount is higher than 2 BNBs, 20% of it will be converted to tokens from the supply and will be sent to a burn address.

The liquidity fee of 4%, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

Future plans

- **SAFUPARD Launchpad**

Launchpad is to be introduced for the upcoming projects to be listed, where investors can find new upcoming projects to give them a unique opportunity to be involved before launch and listing.

- **SAFUPAD Locker**

Any project launched on SAFUPAD Launchpad can lock liquidity through SAFUPAD smart contracts, taking away the ability for those projects to remove or pull liquidity until the end of the set locked period, which allows the SAFUPAD project to stand in a unique use case in the Crypto space, where they can market themselves. Any other project can get the SAFUPAD locker service to lock their liquidity for a certain period.

Potential to grow with score points

1.	Project efficiency	8/10
2.	Project uniqueness	8/10
3	Information quality	8/10
4	Service quality	8/10
5	System quality	8/10
6	Impact on the community	8/10
7	Impact on the business	8/10
8	Preparing for the future	8/10
Total Points		**8/10**

Contract details

Token contract details for 04th July 2021

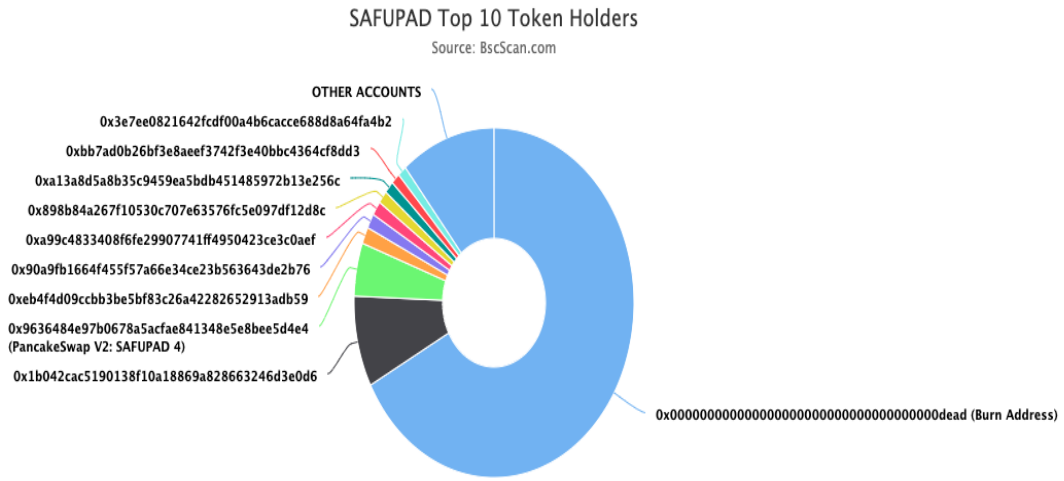
Contract name	SAFUPAD
Contract address	0xEb4f4d09CcBB3be5bf83c26a42282652913aDB59
Token supply	1,000,000,000
Token ticker	SAFUPAD
Decimals	9
Token holders	183
Transaction count	2054
Top 100% holders dominance	99.89%
Contract deployer address	0x75Db9d04bd61EBF6d76B86d191AF7F4D2C787D85
Contract's current owner address	0x75db9d04bd61ebf6d76b86d191af7f4d2c787d85

Top token holders

Top 10 Token Holders

The top 10 holders collectively own 89.07% (890,697,275.70 Tokens) of SAFUPAD

Token Total Supply: 1,000,000,000.00 Token | Total Token Holders: 183

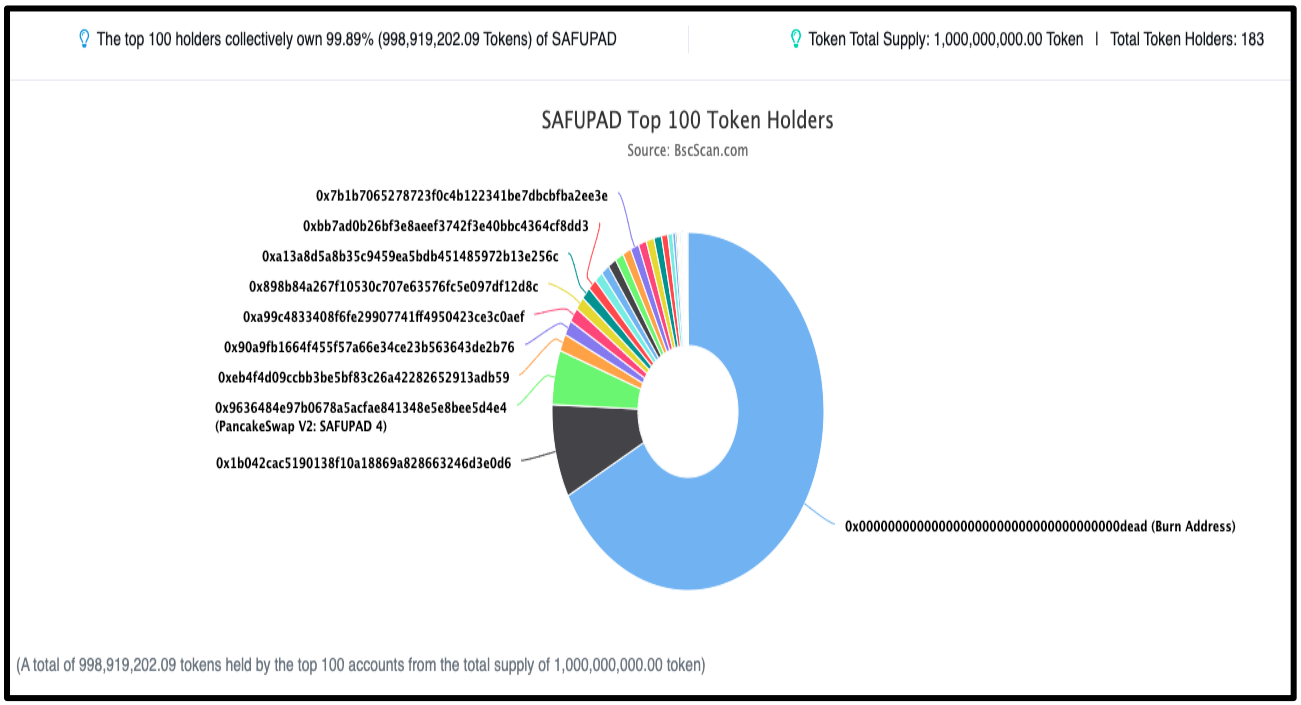


(A total of 890,697,275.70 tokens held by the top 10 accounts from the total supply of 1,000,000,000.00 token)

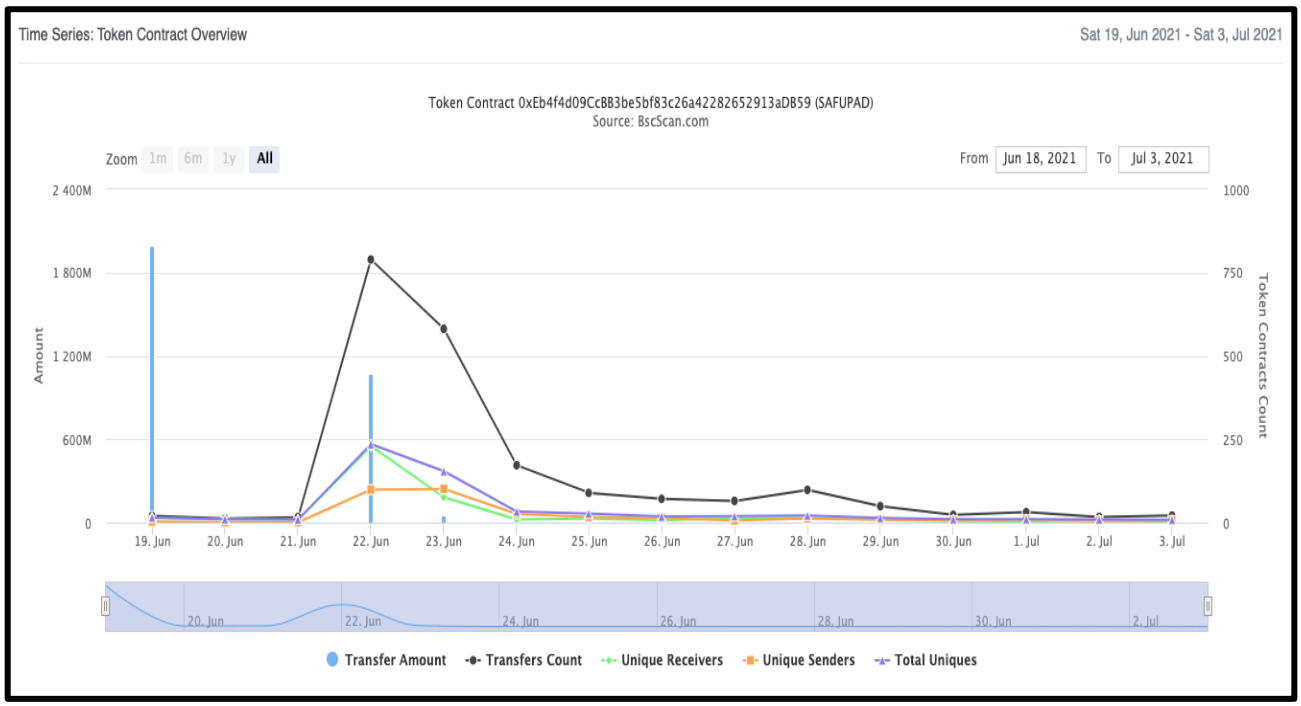
Rank	Address	Quantity (Token)	Percentage
1	Burn Address	672,020,395.729826056	67.2020%
2	0x1b042cac5190138f10a18869a828663246d3e0d6	84,083,488.792247216	8.4083%
3	PancakeSwap V2: SAFUPAD 4	49,215,720.249987425	4.9216%
4	0xeb4f4d09ccbb3be5bf83c26a42282652913adb59	15,401,658.617814065	1.5402%
5	0x90a9fb1664f455f57a66e34ce23b563643de2b76	13,520,740.91527679	1.3521%
6	0xa99c4833408f6fe29907741ff4950423ce3c0aef	13,243,640.287632588	1.3244%
7	0x898b84a267f10530c707e63576fc5e097df12d8c	11,822,756.550655416	1.1823%
8	0xa13a8d5a8b35c9459ea5bdb451485972b13e256c	11,246,450.127940871	1.1246%
9	0xbb7ad0b26bf3e8aeef3742f3e40bbc4364cf8dd3	10,142,424.427680697	1.0142%
10	0x3e7ee0821642fcdf00a4b6cacce688d8a64fa4b2	10,000,000	1.0000%

Token distribution

Top 100 Token Holders



Contract interaction details



Contract code function details

No	Category	Item	Result
1	Coding conventions	BRC20 Token standards	pass
		compile errors	pass
		Compiler version security	pass
		visibility specifiers	pass
		Gas consumption	Low issue
		SafeMath features	pass
		Fallback usage	pass
		tx.origin usage	pass
		deprecated items	pass
		Redundant code	pass
		Overriding variables	pass
2	Function call audit	Authorization of function call	pass
		Low level function (call/delegate call) security	pass
		Returned value security	pass
		Selfdestruct function security	pass
3	Business security	Access control of owners	pass
		Business logics	medium issues
		Business implementations	pass
4	Integer overflow/underflow		pass
5	Reentrancy		pass
6	Exceptional reachable state		pass
7	Transaction ordering dependence		pass
8	Block properties dependence		pass
9	Pseudo random number generator (PRNG)		pass
10	DoS (Denial of Service)		pass
11	Token vesting implementation		pass
12	Fake deposit		pass
13	Event security		pass

Contract description table

Below table represents the summary of the contracts and methods in the token contract. We scanned the whole contract and listed down all the Interfaces, functions and implementations with its visibility and mutability.

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
IBEP20	**Interface**			
L	totalSupply	External ❗		NO❗
L	balanceOf	External ❗		NO❗
L	transfer	External ❗	🛑	NO❗
L	allowance	External ❗		NO❗
L	approve	External ❗	🛑	NO❗
L	transferFrom	External ❗	🛑	NO❗
SafeMath	**Library**			
L	add	Internal 🔒		
L	sub	Internal 🔒		
L	sub	Internal 🔒		
L	mul	Internal 🔒		
L	div	Internal 🔒		
L	div	Internal 🔒		
L	mod	Internal 🔒		
L	mod	Internal 🔒		
Context	**Implementation**			
L	_msgSender	Internal 🔒		

L	_msgData	Internal 🔒		
Address	**Library**			
L	isContract	Internal 🔒		
L	sendValue	Internal 🔒	🛑	
L	functionCall	Internal 🔒	🛑	
L	functionCall	Internal 🔒	🛑	
L	functionCallWithValue	Internal 🔒	🛑	
L	functionCallWithValue	Internal 🔒	🛑	
L	_functionCallWithValue	Private 🔐	🛑	
Ownable	**Implementation**	**Context**		
L		Internal 🔒	🛑	
L	owner	Public ❗		NO❗
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
L	geUnlockTime	Public ❗		NO❗
L	lock	Public ❗	🛑	onlyOwner
L	unlock	Public ❗	🛑	NO❗
IPancakeFactory	**Interface**			
L	feeTo	External ❗		NO❗
L	feeToSetter	External ❗		NO❗
L	getPair	External ❗		NO❗
L	allPairs	External ❗		NO❗

L	allPairsLength	External ❗		NO❗
L	createPair	External ❗	🛑	NO❗
L	setFeeTo	External ❗	🛑	NO❗
L	setFeeToSetter	External ❗	🛑	NO❗
IPancakePair	**Interface**			
L	name	External ❗		NO❗
L	symbol	External ❗		NO❗
L	decimals	External ❗		NO❗
L	totalSupply	External ❗		NO❗
L	balanceOf	External ❗		NO❗
L	allowance	External ❗		NO❗
L	approve	External ❗	🛑	NO❗
L	transfer	External ❗	🛑	NO❗
L	transferFrom	External ❗	🛑	NO❗
L	DOMAIN_SEPARATOR	External ❗		NO❗
L	PERMIT_TYPEHASH	External ❗		NO❗
L	nonces	External ❗		NO❗
L	permit	External ❗	🛑	NO❗
L	MINIMUM_LIQUIDITY	External ❗		NO❗
L	factory	External ❗		NO❗
L	token0	External ❗		NO❗
L	token1	External ❗		NO❗
L	getReserves	External ❗		NO❗
L	price0CumulativeLast	External ❗		NO❗

L	price1CumulativeLast	External ❗		NO❗
L	kLast	External ❗		NO❗
L	mint	External ❗	🛑	NO❗
L	burn	External ❗	🛑	NO❗
L	swap	External ❗	🛑	NO❗
L	skim	External ❗	🛑	NO❗
L	sync	External ❗	🛑	NO❗
L	initialize	External ❗	🛑	NO❗
IPancakeRouter01	**Interface**			
L	factory	External ❗		NO❗
L	WETH	External ❗		NO❗
L	addLiquidity	External ❗	🛑	NO❗
L	addLiquidityETH	External ❗	💵	NO❗
L	removeLiquidity	External ❗	🛑	NO❗
L	removeLiquidityETH	External ❗	🛑	NO❗
L	removeLiquidityWithPermit	External ❗	🛑	NO❗
L	removeLiquidityETHWithPermit	External ❗	🛑	NO❗
L	swapExactTokensForTokens	External ❗	🛑	NO❗
L	swapTokensForExactTokens	External ❗	🛑	NO❗
L	swapExactETHForTokens	External ❗	💵	NO❗
L	swapTokensForExactETH	External ❗	🛑	NO❗

L	swapExactTokensForETH	External ❗	🛑	NO❗
L	swapETHForExactTokens	External ❗	💵	NO❗
L	quote	External ❗		NO❗
L	getAmountOut	External ❗		NO❗
L	getAmountIn	External ❗		NO❗
L	getAmountsOut	External ❗		NO❗
L	getAmountsIn	External ❗		NO❗
IPancakeRouter02	**Interface**	**IPancakeRouter01**		
L	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
Utils	**Library**			
L	random	Private 🔐		
L	isLotteryWon	Private 🔐		

L	calculateBNBReward	Public ❗		NO❗
L	calculateTopUpClaim	Public ❗		NO❗
L	swapTokensForEth	Public ❗	🛑	NO❗
L	swapETHForTokens	Public ❗	🛑	NO❗
L	addLiquidity	Public ❗	🛑	NO❗
ReentrancyGuard	**Implementation**			
L		Public ❗	🛑	NO❗
SAFUPAD	**Implementation**	**Context, IBEP20, Ownable, ReentrancyGuard**		
L		Public ❗	🛑	NO❗
L	name	Public ❗		NO❗
L	symbol	Public ❗		NO❗
L	decimals	Public ❗		NO❗
L	totalSupply	Public ❗		NO❗
L	balanceOf	Public ❗		NO❗
L	transfer	Public ❗	🛑	NO❗
L	allowance	Public ❗		NO❗
L	approve	Public ❗	🛑	NO❗
L	transferFrom	Public ❗	🛑	NO❗
L	increaseAllowance	Public ❗	🛑	NO❗
L	decreaseAllowance	Public ❗	🛑	NO❗
L	isExcludedFromReward	Public ❗		NO❗

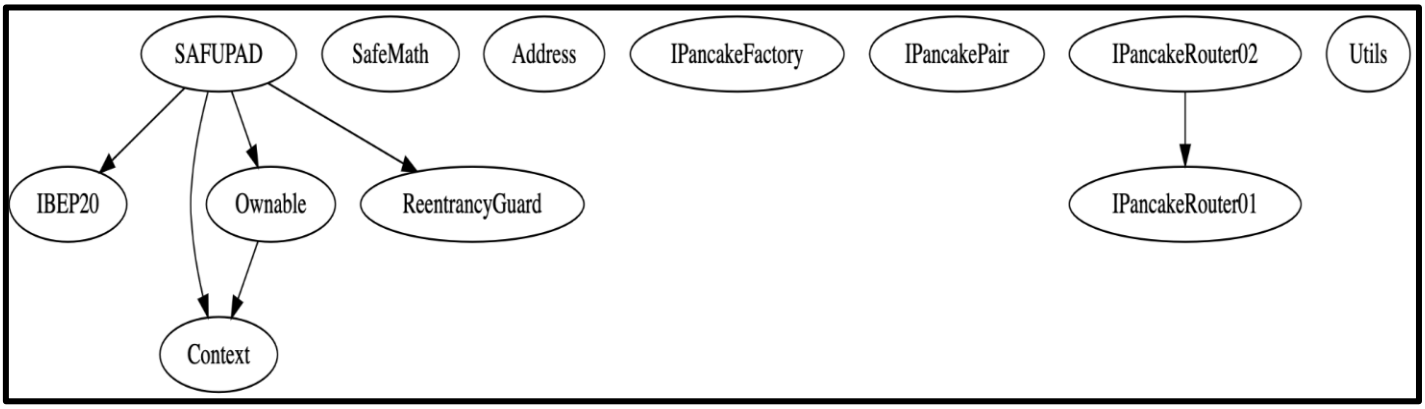
L	totalFees	Public ❗		NO❗
L	deliver	Public ❗	🛑	NO❗
L	reflectionFromToken	Public ❗		NO❗
L	tokenFromReflection	Public ❗		NO❗
L	excludeFromReward	Public ❗	🛑	onlyOwner
L	includeInReward	External ❗	🛑	onlyOwner
L	_transferBothExcluded	Private 🔐	🛑	
L	excludeFromFee	Public ❗	🛑	onlyOwner
L	includeInFee	Public ❗	🛑	onlyOwner
L	setTaxFeePercent	External ❗	🛑	onlyOwner
L	setLiquidityFeePercent	External ❗	🛑	onlyOwner
L	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
L		External ❗	💵	NO❗
L	_reflectFee	Private 🔐	🛑	
L	_getValues	Private 🔐		
L	_getTValues	Private 🔐		
L	_getRValues	Private 🔐		
L	_getRate	Private 🔐		
L	_getCurrentSupply	Private 🔐		
L	_takeLiquidity	Private 🔐	🛑	
L	calculateTaxFee	Private 🔐		
L	calculateLiquidityFee	Private 🔐		
L	removeAllFee	Private 🔐	🛑	

L	restoreAllFee	Private 🔐	🛑	
L	isExcludedFromFee	Public ❗		NO❗
L	_approve	Private 🔐	🛑	
L	_transfer	Private 🔐	🛑	
L	_tokenTransfer	Private 🔐	🛑	
L	_transferStandard	Private 🔐	🛑	
L	_transferToExcluded	Private 🔐	🛑	
L	_transferFromExcluded	Private 🔐	🛑	
L	setMaxTxPercent	Public ❗	🛑	onlyOwner
L	calculateBNBReward	Public ❗		NO❗
L	getRewardCycleBlock	Public ❗		NO❗
L	claimBNBReward	Public ❗	🛑	nonReentrant
L	claimBNBBurnReward	External ❗	🛑	onlyOwner nonReentrant
L	topUpClaimCycleAfterTransfer	Private 🔐	🛑	
L	ensureMaxTxAmount	Private 🔐		
L	disruptiveTransfer	Public ❗	💵	NO❗
L	swapAndLiquify	Private 🔐	🛑	
L	activateContract	Public ❗	🛑	onlyOwner
L	activatePresale	Public ❗	🛑	onlyOwner

Legend

Symbol	Meaning
	Function can modify state
	Function is payable

Inheritance Hierarchy



Security issue checking status

- **High severity issues**
  No high severity issues found

- **Medium severity issues**

  Wrong reward calculation

  When calculating the total supply to get the BNB reward percentage, the contract will get the supply without burn address balance. But when distributing the BNB reward burn address also will get its reward percentage as a normal wallet.

```
ftrace | funcSig
function calculateBNBReward(address ofAddress)
    public
    view
    returns (uint256)
{
    uint256 _totalSupply = uint256(_tTotal).sub(balanceOf(address(0))).sub(
        balanceOf(0x000000000000000000000000000000000000dEaD)
    );

    return
        Utils.calculateBNBReward(
            balanceOf(address(ofAddress)),
            address(this).balance,
            winningDoubleRewardPercentage,
            _totalSupply
        );
}
```

- **Low severity issues**
  No low severity issues found

Owner privileges

(In the period when the owner is not renounced)

❖ Owner can renounce the ownership.

```
    */
    ftrace | funcSig
    function renounceOwnership() public virtual onlyOwner {
        emit OwnershipTransferred(_owner, address(0));
        _owner = address(0);
    }
```

❖ Owner can transfer the ownership.

```
    ftrace | funcSig
    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(
            newOwner != address(0),
            "Ownable: new owner is the zero address"
        );
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }
```

❖ Owner can lock the contract for himself for a certain amount of time period.

```
ftrace | funcSig
function lock(uint256 time) public virtual onlyOwner {
    _previousOwner = _owner;
    _owner = address(0);
    _lockTime = now + time;
    emit OwnershipTransferred(_owner, address(0));
}
```

❖ Owner can unlock the contract when lock time is over.

```
function unlock() public virtual {
    require(
        _previousOwner == msg.sender,
        "You don't have permission to unlock"
    );
    require(now > _lockTime, "Contract is locked until 7 days");
    emit OwnershipTransferred(_owner, _previousOwner);
    _owner = _previousOwner;
}
```

❖ Owner can exclude wallets from reward.

```
ftrace | funcSig
function excludeFromReward(address account) public onlyOwner() {
    // require(account != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not exclude Pancake router.');
    require(!_isExcluded[account], "Account is already excluded");
    if (_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}
```

❖ Owner can include wallets from reward.

```
ftrace | funcSig
function includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

❖ Owner can exclude wallet from all fees.

```
ftrace | funcSig
function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
}
```

❖ Owner can include wallet from all fees.

```
ftrace | funcSig
function includeInFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = false;
}
```

❖ Owner can change the tax fee percentage.

```
ftrace | funcSig
function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    _taxFee = taxFee;
}
```

❖ Owner can change liquidity fee percentage.

```
ftrace | funcSig
function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() {
    _liquidityFee = liquidityFee;
}
```

❖ Owner can enable/disable add liquidity.

```
ftrace | funcSig
function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
    swapAndLiquifyEnabled = _enabled;
    emit SwapAndLiquifyEnabledUpdated(_enabled);
}
```

❖ Owner can send BNB to any wallet that allocated to burn address.

```
function claimBNBBurnReward(address _claimAddress)
    external
    onlyOwner
    nonReentrant
{
    require(
        nextClaimBurnAddress <= block.timestamp,
        "Error: next available not reached"
    );
    require(
        nextAvailableClaimDate[burnAddress] <= block.timestamp,
        "Error: next available not reached"
    );
    require(
        balanceOf(burnAddress) >= 0,
        "Error: must own token to claim reward"
    );

    uint256 reward = calculateBNBReward(burnAddress);

    // reward threshold
    if (reward >= rewardThreshold) {
        Utils.swapETHForTokens(
            address(pancakeRouter),
            address(0x000000000000000000000000000000000000dEaD),
            reward.div(5)
        );
        reward = reward.sub(reward.div(5));
    }

    // update rewardCycleBlock
    nextAvailableClaimDate[burnAddress] =
        block.timestamp +
        getRewardCycleBlock();
    nextClaimBurnAddress = block.timestamp + claimBurnAddressCycle;
    emit ClaimBNBSuccessfully(
        burnAddress,
        reward,
        nextAvailableClaimDate[burnAddress]
    );

    // fixed reentrancy bug
    (bool sent, ) = address(_claimAddress).call{value: reward}("");
    require(sent, "Error: Cannot withdraw reward");
}
```

Special Note

When the holder claims their BNB reward, if their reward exceeds 2 BNB, 20% from their BNB will convert to token and send to the burn address.

```
// reward threshold
if (reward >= rewardThreshold) {
    Utils.swapETHForTokens(
        address(pancakeRouter),
        address(0x000000000000000000000000000000000000dEaD),
        reward.div(5)
    );
    reward = reward.sub(reward.div(5));
}
```

Audit conclusion

While conducting the audit of the SAFUPAD smart contract, it was observed that there is nothing alarming with the code and the contract only contains low and medium severity issues.