auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

SMURFS

DATED : 30 June 23'

AUDIT SUMMARY

Project name – SMURFS

Date: 30 June, 2023

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: **Passed**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	2	0	0
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1- Manual Review:
a line by line code review has been performed by audit ace team.

2- BSC Test Network:
all tests were done on BSC Test network, each test has its transaction has attached to it.

3- Slither : Static Analysis

Testnet Link: all tests were done using this contract, tests are done on BSC Testnet

https://testnet.bscscan.com/token/0x4ffc5e519369b3aed09e7ae9a83b14734beb5a07

Token Information

Token Name : SMURFS

Token Symbol: $SMURFS

Decimals: 18

Token Supply:1,000,000,000,000,000

Token Address:
0xa4Ae624B813Ac08300f3fB0700ffEd373cA55715

Checksum:
0d13ff50475c3fea38371e558f4b13bc5a383542

Owner:
0x542191f552DB0c15F9a4a59403452cA0699aB2b5

TOKEN OVERVIEW

Fees:

Buy Fees: 0-25%

Sell Fees: 0-25%

Transfer Fees: 0-25%

Fees Privilige: Owner

Ownership : Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: none

Blacklist: No

Other Priviliges:

- initial distribution of tokens
- including or excluding from fees
- changing swap threshold
- changing fees

AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.

- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.

- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.

- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.

- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.

- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

auditace.tech

VULNERABILITY CHECKLIST

- Return values of low-level calls
- Private modifier
- Multiple Sends
- Using Suicide
- Gas Limitand Loops
- Address hardcoded
- Exception Disorder
- Using inline assembly
- Divide before multiply
- Missing Zero Address Validation
- Compiler version not fixed
- **Gasless Send**
- Using block.timestamp
- Re-entrancy
- Tautology or contradiction
- Timestamp Dependence
- Revert/require functions
- Use of tx.origin
- Integer overflow/underflow
- Dangerous strict equalities
- Using SHA3
- Using throw

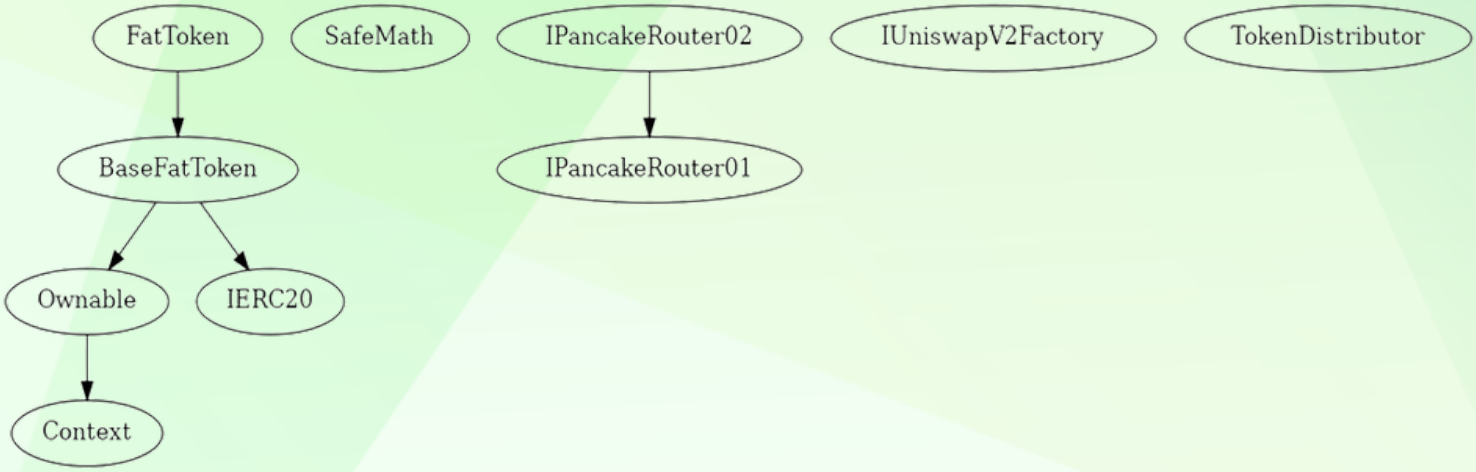
CLASSIFICATION OF RISK

Severity	Description
Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
Gas Optimization /Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity	Found
Critical	0
High-Risk	0
Medium-Risk	2
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is able to set buy/sell/transfer tax up to 25% each
- Owner is not able to blacklist an arbitrary address.
- Owner is not able to set max wallet/transfer/buy/sell
- Owner is not able to mint new tokens

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	owner	Public ❗		NO❗
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
IERC20	Interface			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	decimals	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IPancakeRouter01	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidity	External ❗	🛑	NO❗
└	addLiquidityETH	External ❗	💵	NO❗
IPancakeRouter02	Interface	IPancakeRouter01		

CONTRACT ASSESMENT

└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	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❗
BaseFatToken	Implementation	IERC20, Ownable		
└	setFundAddress	External ❗	🛑	onlyOwner
└	changeSwapLimit	External ❗	🛑	onlyOwner
└	changeWalletLimit	External ❗	🛑	onlyOwner
└	launch	External ❗	🛑	onlyOwner
└	disableSwapLimit	Public ❗	🛑	onlyOwner
└	disableWalletLimit	Public ❗	🛑	onlyOwner
└	disableChangeTax	Public ❗	🛑	onlyOwner
└	completeCustoms	External ❗	🛑	onlyOwner
└	transfer	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	balanceOf	Public ❗		NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	_approve	Private 🔐	🛑	
└	setFeeWhiteList	External ❗	🛑	onlyOwner
└	multi_bclist	Public ❗	🛑	onlyOwner
TokenDistributor	Implementation			
└	<Constructor>	Public ❗	🛑	NO❗
FatToken	Implementation	BaseFatToken		
└	<Constructor>	Public ❗	🛑	NO❗
└	transfer	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	setkb	Public ❗	🛑	onlyOwner
└	isReward	Public ❗		NO❗
└	setAirDropEnable	Public ❗	🛑	onlyOwner
└	_basicTransfer	Internal 🔒	🛑	

CONTRACT ASSESMENT

└	setAirdropNumbs	Public ❗	🛑	onlyOwner
└	setEnableTransferFee	Public ❗	🛑	onlyOwner
└	_transfer	Private 🔐	🛑	
└	setTransferFee	Public ❗	🛑	onlyOwner
└	_tokenTransfer	Private 🔐	🛑	
└	swapTokenForFund	Private 🔐	🛑	lockTheSwap
└	_takeTransfer	Private 🔐	🛑	
└	setSwapPairList	External ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Reentrancy in FatToken._transfer(address,address,uint256) (contracts/Token.sol#489-561):
        External calls:
        - swapTokenForFund(numTokensSellToFund,swapFee) (contracts/Token.sol#544)
                - address(fundAddress).transfer(fundAmount) (contracts/Token.sol#663)
        External calls sending eth:
        - swapTokenForFund(numTokensSellToFund,swapFee) (contracts/Token.sol#544)
                - address(fundAddress).transfer(fundAmount) (contracts/Token.sol#663)
                - _swapRouter.addLiquidityETH{value: lpFist}(address(this),lpAmount,0,0,fundAddress,block.timestamp) (contracts/Token.sol#667-671)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee,isSell,isTransfer) (contracts/Token.sol#560)
                - _balances[to] = _balances[to] + tAmount (contracts/Token.sol#698)
                - _balances[sender] = _balances[sender] - tAmount (contracts/Token.sol#578)
        Event emitted after the call(s):
        - Transfer(sender,to,tAmount) (contracts/Token.sol#699)
                - _tokenTransfer(from,to,amount,takeFee,isSell,isTransfer) (contracts/Token.sol#560)
Reentrancy in FatToken.swapTokenForFund(uint256,uint256) (contracts/Token.sol#627-695):
        External calls:
        - address(fundAddress).transfer(fundAmount) (contracts/Token.sol#663)
        External calls sending eth:
        - address(fundAddress).transfer(fundAmount) (contracts/Token.sol#663)
        - _swapRouter.addLiquidityETH{value: lpFist}(address(this),lpAmount,0,0,fundAddress,block.timestamp) (contracts/Token.sol#667-671)
        Event emitted after the call(s):
        - Failed_AddLiquidityETH() (contracts/Token.sol#670)
Reentrancy in FatToken.transferFrom(address,address,uint256) (contracts/Token.sol#438-444):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#439)
                - address(fundAddress).transfer(fundAmount) (contracts/Token.sol#663)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#439)
                - address(fundAddress).transfer(fundAmount) (contracts/Token.sol#663)
                - _swapRouter.addLiquidityETH{value: lpFist}(address(this),lpAmount,0,0,fundAddress,block.timestamp) (contracts/Token.sol#667-671)
        State variables written after the call(s):
        - _allowances[sender][msg.sender] = _allowances[sender][msg.sender] - amount (contracts/Token.sol#441)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#158) is too simil
ar to IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

BaseFatToken.deadAddress (contracts/Token.sol#247) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

BaseFatToken._mainPair (contracts/Token.sol#258) should be immutable
BaseFatToken._swapRouter (contracts/Token.sol#254) should be immutable
BaseFatToken.currency (contracts/Token.sol#225) should be immutable
BaseFatToken.currencyIsEth (contracts/Token.sol#215) should be immutable
BaseFatToken.decimals (contracts/Token.sol#244) should be immutable
BaseFatToken.enableKillBlock (contracts/Token.sol#218) should be immutable
BaseFatToken.enableOffTrade (contracts/Token.sol#217) should be immutable
BaseFatToken.enableRewardList (contracts/Token.sol#219) should be immutable
BaseFatToken.name (contracts/Token.sol#242) should be immutable
BaseFatToken.symbol (contracts/Token.sol#243) should be immutable
BaseFatToken.totalSupply (contracts/Token.sol#245) should be immutable
FatToken._tokenDistributor (contracts/Token.sol#352) should be immutable
FatToken.enableTransferFee (contracts/Token.sol#478) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

FUNCTIONAL TESTING

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

All the functionalities have been tested, no issues were found

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x6a6a6d1f515719fa0313df0c579763113434fb618b41dd2d59fe250213a1040d

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xa49287a23121820961659e420b4bc8dc829046d65f0dfbcea6c2b52497bbd63f

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xf3be19749fcab96954b855eba420a0eb2370f30b5147fe2b60d02cb3c8de5e87

4- Transferring when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x561ff3cdcbf43a8cdf46e3e21bbfe10a08816b92e43142709c0b5cb50a2a3c66

5- Buying when not excluded from fees (0-25% tax) (passed):
https://testnet.bscscan.com/tx/0x158dd2e133048d5323158eb07732451d7e7a53956f7522b0395a91cab27ed4ac

6- Selling when not excluded from fees (0-25% tax) (passed):
https://testnet.bscscan.com/tx/0x9040436b967b1be2b66468cbb042642ec5c0381e52bfca68437e7e8f4dfd0fa3

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (0-25% tax) (passed):
https://testnet.bscscan.com/tx/0x8198bce9f975ff97b6b72db942d4ddf19e306218bebf47c5b099e6f06efb3547

8-Internal swap (passed):
-Auto-liquidity
- Fund wallet received BNB
https://testnet.bscscan.com/tx/0x9040436b967b1be2b66468cbb042642ec5c0381e52bfca68437e7e8f4dfd0fa3

9- Airdrop (passed):
https://testnet.bscscan.com/tx/0x8198bce9f975ff97b6b72db942d4ddf19e306218bebf47c5b099e6f06efb3547

ISSUES FOUND

Centralization – Excessive fees

Severity: Medium

function: completeCustoms

Status: Not Resolved

Overview:

Owner is able to set 25% tax on buy/sell/transfers.

```
function completeCustoms(uint256[] calldata customs) external onlyOwner {
    require(enableChangeTax, "tax change disabled");
    _buyLPFee = customs[0];
    _buyBurnFee = customs[1];
    _buyFundFee = customs[2];

    _sellLPFee = customs[3];
    _sellBurnFee = customs[4];
    _sellFundFee = customs[5];

    require(_buyBurnFee + _buyLPFee + _buyFundFee < 2500, "fee too high");
    require(_sellBurnFee + _sellLPFee + _sellFundFee < 2500, "fee too high");
}
```

High amount of buy/sell fees can often be used to control price volatility / buy – sell pressure at launch time, however this is still considered a centralization issue, because:

1- buy/sell/transfer fees are exceeding the safe range (0-10% according to pinksale safu criteria)

2- This function can be called anytime meaning that its not limited to launch time

Suggestion

Its suggested to keep fees in range 0-10% for buy/sell/transfers

```
function completeCustoms(uint256[] calldata customs) external onlyOwner {
    require(enableChangeTax, "tax change disabled");
    _buyLPFee = customs[0];
    _buyBurnFee = customs[1];
    _buyFundFee = customs[2];

    _sellLPFee = customs[3];
    _sellBurnFee = customs[4];
    _sellFundFee = customs[5];

    require(_buyBurnFee + _buyLPFee + _buyFundFee < 1000, "fee too high");
    require(_sellBurnFee + _sellLPFee + _sellFundFee < 1000, "fee too high");
}
```

ISSUES FOUND

Centralization – EOA receiving LP

Severity: Medium

Status: Not Resolved

Overview:

an EOA is receiving LP tokens generated from auto-liquidty. This tokens can be used to remove a portion of liquidity pool.

```
if (lpAmount > 0 && lpFist > 0) {
    // add the liquidity
    try _swapRouter.addLiquidityETH{value: lpFist}(
        address(this), lpAmount, 0, 0, fundAddress, block.timestamp
    ) {} catch {
        emit Failed_AddLiquidityETH();
    }
}
```

Suggestion

Its suggested to Burn or lock new LP tokens

DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.

ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.

https://auditace.tech/

https://t.me/Audit_Ace

https://twitter.com/auditace_

https://github.com/Audit-Ace