auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

XFloki

DATED : 31 July 23'

High Risk

Centralization – Trades are disabled by default

Severity: High

function: launch

Status: Not Resolved

Overview:

Owner must enable trades manually, otherwise holders wont be able to buy/sell/transfer tokens.

```
function launch() external onlyOwner {
    require(startTradeBlock == 0, "already started");
    startTradeBlock = block.number;
}
```

Suggestion

Its suggested to either enable trades prior to presale, or transfer ownership of the contract to a trusted 3rd party like a certified pinksale safu developer.

High Risk

Centralization – Maximum buy and sell

Severity: High

function: launch

Status: Not Resolved

Overview:

Owner is able to set maximum buy or sell amount each to zero.

```
    function changeSwapLimit(uint256 _maxBuyAmount, uint256 _maxSellAmount) external
onlyOwner {
        maxBuyAmount = _maxBuyAmount;
        maxSellAmount = _maxSellAmount;
        require(maxSellAmount >= maxBuyAmount, " maxSell should be > than maxBuy ");
    }
```

Suggestion

Set an upper bound for maximum amount of buy and sell limits.

```
    function changeSwapLimit(uint256 _maxBuyAmount, uint256 _maxSellAmount) external
onlyOwner {
        maxBuyAmount = _maxBuyAmount;
        maxSellAmount = _maxSellAmount;
        reuqire(maxBuyAmount >= totalSupply() / 1000, “Can’t set mximum buy lower than 0.1% of
supply”);
        reuqire(maxSellAmount >= totalSupply() / 1000, “Can’t set mximum sell lower than 0.1% of
supply”));
        require(maxSellAmount >= maxBuyAmount, " maxSell should be > than maxBuy ");
    }
```

AUDIT SUMMARY

Project name – XFloki

Date: 31 July, 2023

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

Audit Status: Passed With High Risk

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	2	1	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 conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

3- Slither :
The code has undergone static analysis using Slither.

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:
https://testnet.bscscan.com/token/0x06d49Eb9217603857E03f47c618C5BfF9dEF249B

Token Information

Token Name : XFloki

Token Symbol: XFLOKI

Decimals: 18

Token Supply: 69,420

Token Address:
0x2C4f9461Fc727c3A4e3116102BF4B064FC9aFdBB

Checksum:
0d13ff50475c3fea38371e558f4b13bc5a383542

Owner:
0x42C813E8463D63FA1CbE823a47438B503fb48A19
(at time of writing the audit)

Deployer:
0xCbbB74c36De813C3F39347f3dD5A29323BbF26cE

TOKEN OVERVIEW

Fees:

Buy Fees: 0-25%

Sell Fees: 0-25%

Transfer Fees: 0%

Fees Privilege: Owner

Ownership: owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: Initial distribution of the tokens

modifying fees

disabling trades

enabling trades

setting maximum buy/sell amounts

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.

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

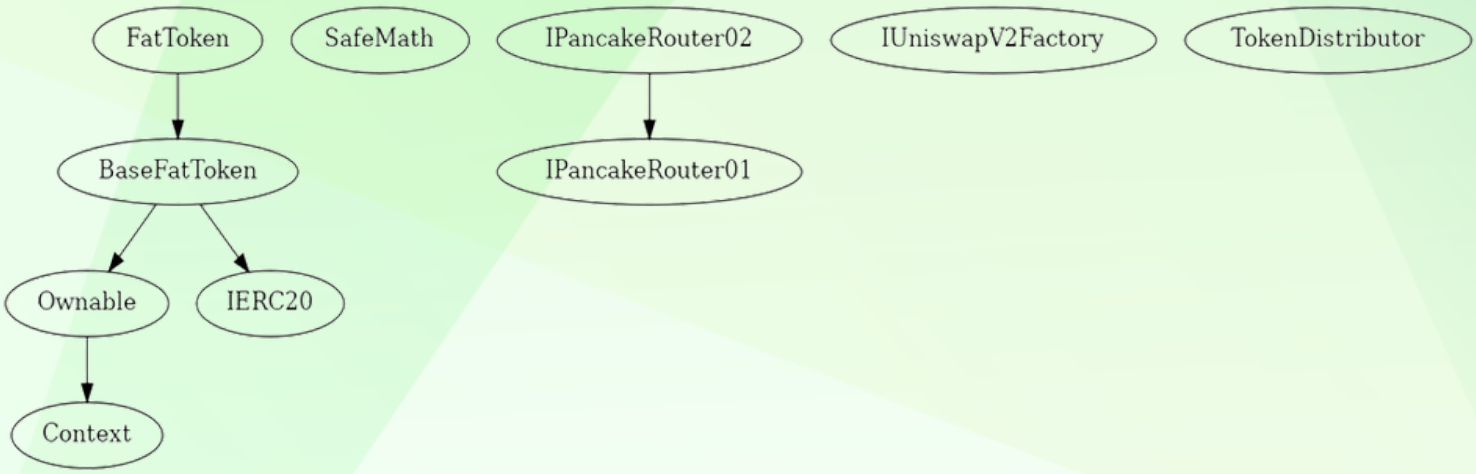
CLASSIFICATION OF RISK

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	2
Medium-Risk	1
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is able to change current fees within 0-25% for buy and sells
- Owner is not able to set buy on transfsers
- Owner is not able to blacklist an arbitrary address.
- **Owner is able to set max buy and sell**
- Owner is not able to mint new tokens
- **Owner must enable trades manually**
- **Owner is able to disable trades**

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		
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗

CONTRACT ASSESMENT

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 🔒		🛑		
└	setAirdropNumbs	Public ❗		🛑	onlyOwner
└	setEnableTransferFee	Public ❗		🛑	onlyOwner
└	_transfer	Private 🔐		🛑		

CONTRACT ASSESMENT

└	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

1- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0xb01a33a8952fc3194577d8855a4db9b71619b5275303274cbc1ba272f2cf037b

2- Buying when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0x7d8720ac107472417319281ed5ed203b23a2dc21cecc6d15769b5dc34ce5878b

3- Selling when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0xc4f9a6be88f8d637d8d5f603a3f924b43f588055a9a23197b3588a1ceb8de4fb

4- Transferring when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0xbdd71ac7d8c6be6d7184579ae7d1d1534df83eef9b76d1633151ea96f71e5c93

5- Buying when not excluded from fees (0-25% tax) (passed):

https://testnet.bscscan.com/tx/0xd2d3dff9bd9249be4c52bba447729ef254d148b1e13cf6880ec39147fdc424b6

FUNCTIONAL TESTING

6- Selling when not excluded from fees (0-25% tax) (passed):

https://testnet.bscscan.com/tx/0x49b014606d8dc8ea6a679fe965f19df8296b10728e4e21b30e52a46983246522

7- Transferring when not excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0x08bb241c0ac7c3637d35f6b69c997bae4db6ec1fa09568468d52949478e84fd5

8- Internal swap (passed):

https://testnet.bscscan.com/tx/0x49b014606d8dc8ea6a679fe965f19df8296b10728e4e21b30e52a46983246522

High Risk

Centralization – Trades are disabled by default

Severity: High

function: launch

Status: Not Resolved

Overview:

Owner must enable trades manually, otherwise holders wont be able to buy/sell/transfer tokens.

```
function launch() external onlyOwner {
    require(startTradeBlock == 0, "already started");
    startTradeBlock = block.number;
}
```

Suggestion

Its suggested to either enable trades prior to presale, or transfer ownership of the contract to a trusted 3rd party like a certified pinksale safu developer.

High Risk

Centralization – Maximum buy and sell

Severity: High

function: launch

Status: Not Resolved

Overview:

Owner is able to set maximum buy or sell amount each to zero.

```
    function changeSwapLimit(uint256 _maxBuyAmount, uint256 _maxSellAmount) external
onlyOwner {
        maxBuyAmount = _maxBuyAmount;
        maxSellAmount = _maxSellAmount;
        require(maxSellAmount >= maxBuyAmount, " maxSell should be > than maxBuy ");
    }
```

Suggestion

Set an upper bound for maximum amount of buy and sell limits.

```
    function changeSwapLimit(uint256 _maxBuyAmount, uint256 _maxSellAmount) external
onlyOwner {
        maxBuyAmount = _maxBuyAmount;
        maxSellAmount = _maxSellAmount;
        reuqire(maxBuyAmount >= totalSupply() / 1000, “Can’t set mximum buy lower than 0.1% of
supply”);
        reuqire(maxSellAmount >= totalSupply() / 1000, “Can’t set mximum sell lower than 0.1% of
supply”));
        require(maxSellAmount >= maxBuyAmount, " maxSell should be > than maxBuy ");
    }
```

Medium Risk

Logical – Lost funds

Severity: Medium

function: ----

Status: Not Resolved

Overview:

There are no functions to withdraw ERC20 tokens or BNB from the contract. All the tokens sent to the contract will be locked forever.

Suggestion

Its strongly suggested to implement two functions for withdrawing ERC20 tokens and BNB from the contract.

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