Smart Contract Vulnerability Audit

Purple Floki

Dec 05, 2021

Smart Contract – Audit Overview

Project Summary

Project Name	Purple Floki
Platform	Binance Smart Chain
Language	Solidity
Commits	0x83713F2219894deb559Ec5530e80e3bC116334dD

Audit Summary

Delivery Date	December 06, 2021
Method of Audit	Human and AI
Consultants Engaged	Two
Timeline	December 05, 2021 – December 07, 2021

Vulnerability Summary

Vulnerability Level	Total	Resolved
Critical	0	✓
Major	0	✓
Medium	0	✓
Minor	0	✓
Informational	4	X

Smart Contract - Contract Overview

All information is recorded as of 12/05/2021

Contract Name	PURPLEFLOKI.sol
Contract Ticker	PURPLEFLOKI
Contract Address	0x83713F2219894deb559Ec5530e80e3bC116334dD
Contract Creator	0xb1E856636284Fbbfc690605dBcff5D57662e6529
Decimals	9
Total Supply	1,000,000,000,000,000
Token Holders	22
Token Transfers	30
Complier Version	v0.6.12+commit.27d51765
Source Code	Solidity
Optimization Enabled	No with 200 runs
Other Settings	default evmVersion, None license

Smart Contract - Vulnerabilities

Vulnerability Tested	Human Review	Ai Review	Line(s) Affected	Results
Function Default Visibility	✓	✓		✓
Integer Overflow and Underflow	✓	✓		✓
Outdated Compiler Version	✓	✓		✓
Floating Pragma	✓	✓	L5	✗
Unchecked Call Return Value	✓	✓		✓
Unprotected Ether Withdrawal	✓	✓		✓
Unprotected SELFDESTRUCT Instruction	✓	✓		✓
Unencrypted Private Data On-Chain	✓	✓		✓

Smart Contract - Vulnerabilities

Vulnerability Tested	Human Review	Ai Review	Line(s) Affected	Results
Reentrancy	✓	✓		✓
State Variable Default Visibility	✓	✓	L723, L730, L737	✗
Uninitialized Storage Pointer	✓	✓		✓
Assert Violation	✓	✓		✓
Use of Deprecated Solidity Functions	✓	✓		✓
Delegatecall to Untrusted Callee	✓	✓		✓
DoS with Failed Call	✓	✓		✓
Code With No Effects	✓	✓		✓

Smart Contract - Vulnerabilities

Vulnerability Tested	Human Review	Ai Review	Line(s) Affected	Results
Transaction Order Dependence	✓	✓		✓
Authorization through tx.origin	✓	✓		✓
Block values as a proxy for time	✓	✓		✓
Signature Malleability	✓	✓		✓
Incorrect Constructor Name	✓	✓		✓
Shadowing State Variables	✓	✓		✓
Weak Sources of Randomness from Chain Attributes	✓	✓		✓

Smart Contract - Vulnerabilities

Vulnerability Tested	Human Review	Ai Review	Line(s) Affected	Results
Missing Protection against Signature Replay Attacks	✓	✓		✓
Lack of Proper Signature Verification	✓	✓		✓
Requirement Violation	✓	✓		✓
Write to Arbitrary Storage Location	✓	✓		✓
Incorrect Inheritance Order	✓	✓		✓
Insufficient Gas Griefing	✓	✓		✓
Arbitrary Jump with Function Type Variable	✓	✓		✓

Smart Contract - Vulnerabilities

Vulnerability Tested	Human Review	Ai Review	Line(s) Affected	Results
DoS With Block Gas Limit	✓	✓		✓
Typographical Error	✓	✓		✓
Right-To-Left-Override control character	✓	✓		✓
Presence of unused variables	✓	✓		✓
Unexpected Ether balance	✓	✓		✓
Hash Collisions With Multiple Variable Length Arguments	✓	✓		✓
Message call with hardcoded gas amount	✓	✓		✓

Smart Contract - Code Analysis

Floating Pragma
Severity: Informational
PURPLEFLOKI.sol
Line: 5

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.

```
3   */
4
5   pragma solidity ^0.6.12;
6   // SPDX-License-Identifier: Unlicensed
7   interface IERC20 {
```

State variable visibility is not set
Severity: Informational
PURPLEFLOKI.sol
Line: 723

It is best practice to set the visibility of state variables explicitly. The default visibility for "buyFeesActive" is internal. Other possible visibility settings are public and private.

```
721 uint256 private _previousLiquidityFee = _liquidityFee;
722
723 bool buyFeesActive = false;
724
725 address public marketingWallet;
```

State variable visibility is not set
Severity: Informational
PURPLEFLOKI.sol
Line: 730

It is best practice to set the visibility of state variables explicitly. The default visibility for "inSwapAndLiquify" is internal. Other possible visibility settings are public and private.

```
728 address public immutable uniswapV2Pair;
729
730 bool inSwapAndLiquify;
731 bool public swapAndLiquifyEnabled = true;
732
```

State variable visibility is not set
Severity: Informational
PURPLEFLOKI.sol
Line: 737

It is best practice to set the visibility of state variables explicitly. The default visibility for "operator" is internal. Other possible visibility settings are public and private.

```
735 uint256 private numTokensSellToAddToLiquidity = 1000000000000 * 10**9; // 0.1%
736
737 address operator;
738
739 event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
```

Smart Contract - Contract Functions

+ [Int] IERC20
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] transfer #
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transferFrom #

+ [Lib] SafeMath
  - [Int] add
  - [Int] sub
  - [Int] sub
  - [Int] mul
  - [Int] div
  - [Int] div
  - [Int] mod
  - [Int] mod

+ Context
  - [Int] _msgSender
  - [Int] _msgData

+ [Lib] Address
  - [Int] isContract
  - [Int] sendValue #
  - [Int] functionCall #
  - [Int] functionCall #
  - [Int] functionCallWithValue #
  - [Int] functionCallWithValue #
  - [Prv] _functionCallWithValue #

+ Ownable (Context)
  - [Int] <Constructor> #
  - [Pub] owner
  - [Pub] renounceOwnership #
    - modifiers: onlyOwner
  - [Pub] transferOwnership #
    - modifiers: onlyOwner
  - [Pub] geUnlockTime
  - [Pub] lock #
    - modifiers: onlyOwner
  - [Pub] unlock #

Smart Contract - Contract Functions

+ [Int] IUniswapV2Factory
  - [Ext] feeTo
  - [Ext] feeToSetter
  - [Ext] getPair
  - [Ext] allPairs
  - [Ext] allPairsLength
  - [Ext] createPair #
  - [Ext] setFeeTo #
  - [Ext] setFeeToSetter #

+ [Int] IUniswapV2Pair
  - [Ext] name
  - [Ext] symbol
  - [Ext] decimals
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transfer #
  - [Ext] transferFrom #
  - [Ext] DOMAIN_SEPARATOR
  - [Ext] PERMIT_TYPEHASH
  - [Ext] nonces
  - [Ext] permit #
  - [Ext] MINIMUM_LIQUIDITY
  - [Ext] factory
  - [Ext] token0
  - [Ext] token1
  - [Ext] getReserves
  - [Ext] price0CumulativeLast
  - [Ext] price1CumulativeLast
  - [Ext] kLast
  - [Ext] mint #
  - [Ext] burn #
  - [Ext] swap #
  - [Ext] skim #
  - [Ext] sync #
  - [Ext] initialize #

Smart Contract - Contract Functions

+ [Int] IUniswapV2Router01
  - [Ext] factory
  - [Ext] WETH
  - [Ext] addLiquidity #
  - [Ext] addLiquidityETH ($)
  - [Ext] removeLiquidity #
  - [Ext] removeLiquidityETH #
  - [Ext] removeLiquidityWithPermit #
  - [Ext] removeLiquidityETHWithPermit #
  - [Ext] swapExactTokensForTokens #
  - [Ext] swapTokensForExactTokens #
  - [Ext] swapExactETHForTokens ($)
  - [Ext] swapTokensForExactETH #
  - [Ext] swapExactTokensForETH #
  - [Ext] swapETHForExactTokens ($)
  - [Ext] quote
  - [Ext] getAmountOut
  - [Ext] getAmountIn
  - [Ext] getAmountsOut
  - [Ext] getAmountsIn

+ [Int] IUniswapV2Router02 (IUniswapV2Router01)
  - [Ext] removeLiquidityETHSupportingFeeOnTransferTokens #
  - [Ext] removeLiquidityETHWithPermitSupportingFeeOnTransferTokens #
  - [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
  - [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens ($)
  - [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+ PURPLEFLOKI (Context, IERC20, Ownable)
  - [Pub] <Constructor> #
  - [Ext] changeOperator #
    - modifiers: onlyOperator
  - [Pub] name
  - [Pub] symbol
  - [Pub] decimals
  - [Pub] totalSupply
  - [Pub] balanceOf
  - [Pub] transfer #
  - [Pub] allowance
  - [Pub] approve #
  - [Pub] transferFrom #
  - [Pub] increaseAllowance #
  - [Pub] decreaseAllowance #
  - [Pub] isExcludedFromReward
  - [Pub] totalFees
  - [Pub] deliver #
  - [Pub] reflectionFromToken
  - [Pub] tokenFromReflection
  - [Pub] excludeFromReward #
    - modifiers: onlyOwner

Smart Contract - Contract Functions

- [Ext] includeInReward #
  - modifiers: onlyOwner
- [Prv] _transferBothExcluded #
- [Pub] excludeFromFee #
  - modifiers: onlyOperator
- [Pub] includeInFee #
  - modifiers: onlyOperator
- [Ext] setTaxFeePercent #
  - modifiers: onlyOwner
- [Ext] setLiquidityFeePercent #
  - modifiers: onlyOwner
- [Ext] setBurnFeePercent #
  - modifiers: onlyOwner
- [Ext] setMarketingFeePercent #
  - modifiers: onlyOwner
- [Ext] setMaxTxPercent #
  - modifiers: onlyOwner
- [Ext] setMaxWalletPercent #
  - modifiers: onlyOwner
- [Pub] setSwapAndLiquifyEnabled #
  - modifiers: onlyOwner
- [Ext] <Receive Ether> ($)
- [Prv] _reflectFee #
- [Prv] _getValues
- [Prv] _getTValues
- [Prv] _getRValues
- [Prv] _getRate
- [Prv] _getCurrentSupply
- [Prv] _takeLiquidity #
- [Prv] calculateTaxFee
- [Prv] calculateBurnFee
- [Prv] calculateMarketingFee
- [Prv] calculateLiquidityFee
- [Prv] removeAllFee #
- [Prv] restoreAllFee #
- [Pub] isExcludedFromFee
- [Prv] _approve #
- [Prv] _transfer #
- [Prv] swapAndLiquify #
  - modifiers: lockTheSwap
- [Prv] swapTokensForEth #
- [Prv] addLiquidity #
- [Prv] _tokenTransfer #
- [Prv] _transferStandard #
- [Prv] _transferToExcluded #
- [Prv] _transferFromExcluded #
- [Ext] withdrawStuckBNB #
  - modifiers: onlyOperator
- [Ext] removeStuckToken #
  - modifiers: onlyOperator

Smart Contract – Owner Functions

- Owner can modify tax fee percent
- Owner can modify liquidity fee
- Owner can modify burn fee
- Owner can modify marketing fee
- Owner can set max tx
- Owner can set max wallet
- Owner can enable liquidity
- Owner can remove stuck tokens
- Owner can whitelist addresses from fee
- Owner can include addresses for fee
- Owner can include addresses for rewards
- Owner can whitelist addresses from rewards
- Owner can set tax fee
- Owner can withdraw stuck BNB

Smart Contract – Owner Functions

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

function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() {
    _liquidityFee = liquidityFee;
}

function setBurnFeePercent(uint256 burnFee) external onlyOwner() {
    _burnFee = burnFee;
}

 function setMarketingFeePercent(uint256 fee) external onlyOwner() {
     _marketingFee = fee;
 }

function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner() {
    _maxTxAmount = _tTotal.mul(maxTxPercent).div(
        10**2
    );
}

function setMaxWalletPercent(uint256 maxWallPercent) external onlyOwner() {
    _maxWalletSize = _tTotal.mul(maxWallPercent).div(
        10**2
    );

function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
    swapAndLiquifyEnabled = _enabled;
    emit SwapAndLiquifyEnabledUpdated(_enabled);
}

 //to recieve ETH from uniswapV2Router when swaping

function removeStuckToken(address _address) external onlyOperator {
    require(_address != address(this), "Can't withdraw tokens destined for liquidity");
    require(IERC20(_address).balanceOf(address(this)) > 0, "Can't withdraw 0");

    IERC20(_address).transfer(owner(), IERC20(_address).balanceOf(address(this)));
}
```

Smart Contract – Owner Functions

```
function excludeFromFee(address account) public onlyOperator {
    _isExcludedFromFee[account] = true;
}

function includeInFee(address account) public onlyOperator {
    _isExcludedFromFee[account] = false;
}

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;
        }
    }
}

function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    _taxFee = taxFee;
}

function withdrawStuckBNB() external onlyOperator{
    require (address(this).balance > 0, "Can't withdraw negative or zero");
    payable(owner()).transfer(address(this).balance);
}

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

Smart Contract –
Mint function

This contract does not contain a mint function. We were unable to locate a mint function within the code.

Smart Contract –
Contract Ownership

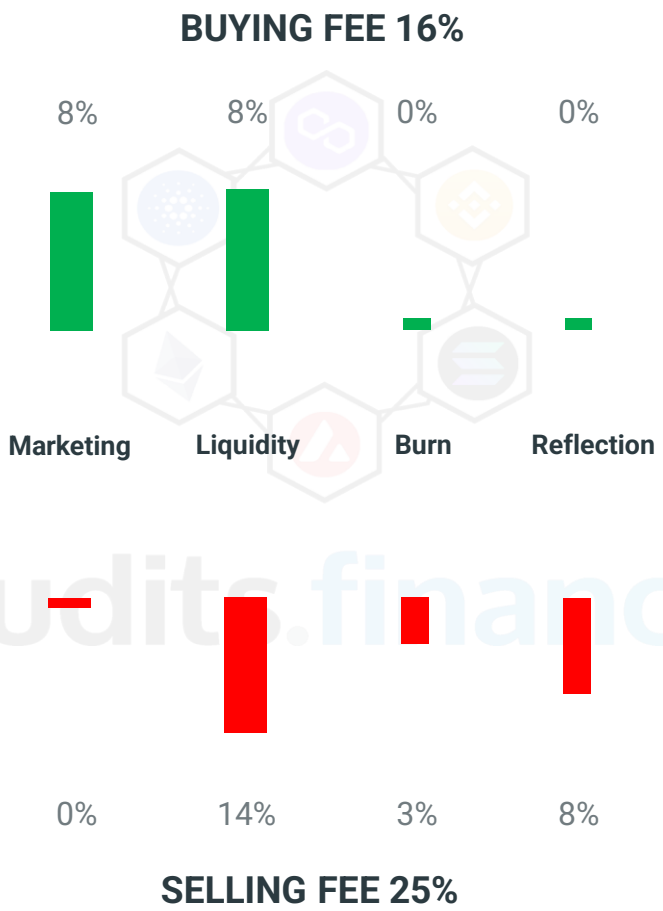
Contract ownership has not been renounced at the time of the audit.
The owner's address is shown as:
0xb1e856636284fbbfc690605dbcff5d57662e6529

Smart Contract –
Locked Liquidity

Locked liquidity information was not identified at the time of the audit completion. Locked liquidity is always be subjected to change.

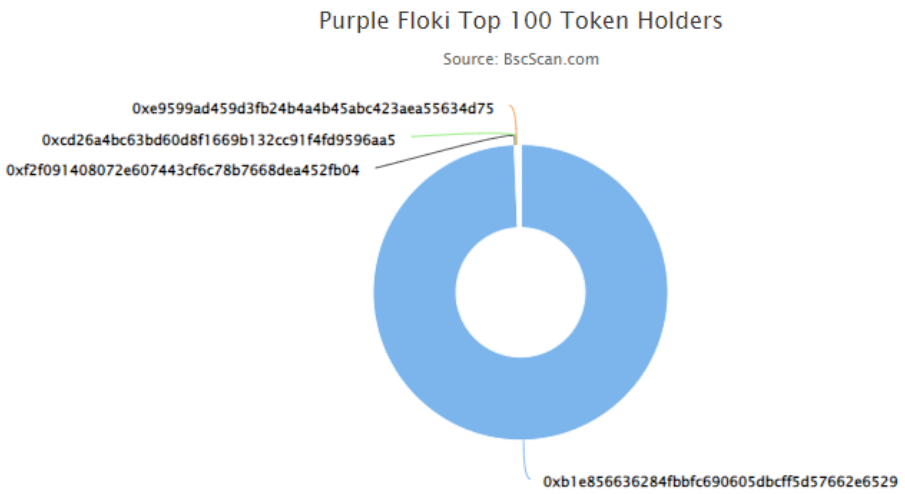
Smart Contract - Tokenomics

At the time of audit the transaction fees ("tax") listed below are the fees associated with trading. These fees are taken from every buy and sell transaction unless otherwise stated. Token taxes vary by each project.

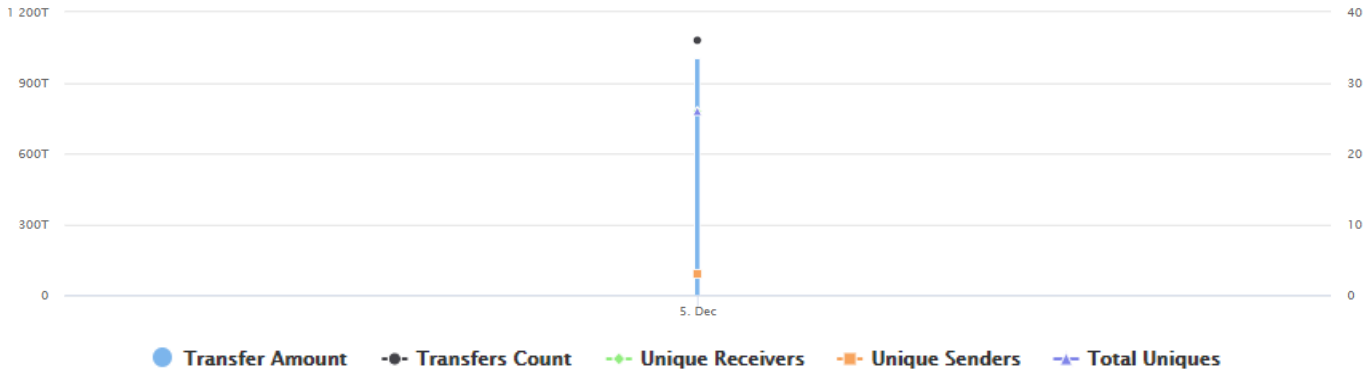


Token Holders & Contract Analytics

Top 100 Token Holders



Token Contract Analytics



Team Overview

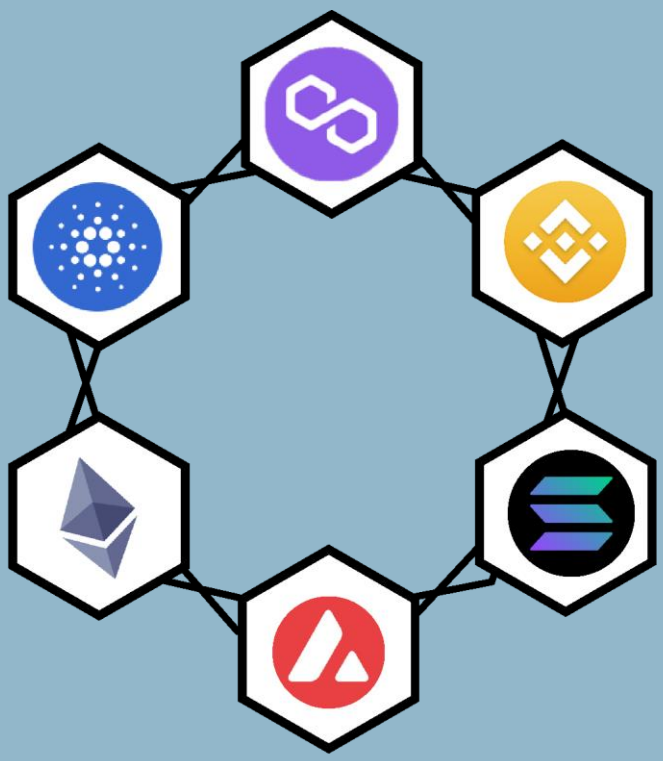
audits.finance

KYC NOT CERTIFIED

Audits.finance has not completed a KYC for the project. Audits.finance has not verified the identity of any team member(s) with government issued ID and photo evidence to match. This project is anonymous.

DISCLAIMER

Audits.finance Inc. is in no way responsible or liable for any legal actions resulting from the use of this presentation. By reading this audit or any part of it, you agree to the terms of this disclaimer. If you do not agree to these terms, please stop reading now, and delete any duplicates of this report. Audits.finance Inc. is an official auditor utilizing the Solidity auditing industry standard. Audits.finance hereby excludes any liability and responsibility. Neither you nor any other person shall have any claim against Audits.finance for any economic loss or damages. Audits.finance Inc. does not guarantee the authenticity of a project, nor does it guarantee the project will not participate in one or any scamming including but not limited to, removing liquidity, selling off team supply, or exit scams. Audits.finance Inc. does not give investment advice in any way. Audits.finance Inc. supplies this presentation for information purposes only, and strongly suggests that none of this information be used as investment advice. Audits.finance in no way endorses or recommends any projects that it audits. Audits.finance is solely responsible for smart contract and project analysis of the projects that it is contracted to audit. Audits.finance may be contracted by teams, investors, or any other 3rd party in regard to a contract address or project. Audits.finance provides a full report for informational purposes only.



audits.finance

Contact information:
Website: audits.finance
Telegram: auditsfinancegroup
Twitter: auditsfinance
Email: hello@audits.finance