Audit Report for **UP611**
Date: 13 June 2024
Audit result: **High-Risk Major Flag**

Token Address: 0xAC66008B0a72048c048cC1F766e76D46e4F247cB

Name: UP611

Symbol: UP611

Decimals: 18

Network: BscScan

Token Type: BEP-20

Owner: -

Deployer: 0x6De47C44E99b02519e7Fa34daF730d4f691b1aee

Token Supply: 500000000

Checksum: A17acbefe2a12642d388659dffd20211

Testnet:
https://testnet.bscscan.com/address/0xf27f6ac1239e1466a38dfd4b2cf40431a7ab7a58#code

Token Overview:

Buy Fee: 0%

Sell Fee: 0%

Transfer Fee: 0%

Fee Privilege: Owner

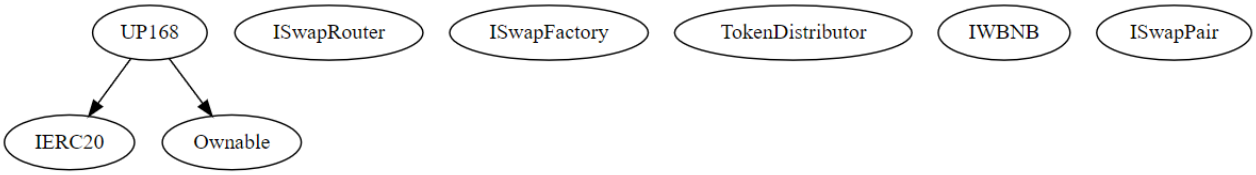
Ownership: Owned

Minting: None

Max Tx: No

Blacklist: Yes

Inheritance Tree



Static Analysis

A static analysis of the code was performed using Slither. No issues were found.

```
INFO:Detectors:
UP168.swapTokenForFund(uint256,uint256) (UP168.sol#734-826) performs a multiplication on the result of a division:
        - toFundAmt = (newBal * (_buyFundFee + _sellFundFee)) / totalShare (UP168.sol#771-772)
        - amountOfFund = toFundAmt * 1 / 13 (UP168.sol#780)
UP168.swapTokenForFund(uint256,uint256) (UP168.sol#734-826) performs a multiplication on the result of a division:
        - toFundAmt = (newBal * (_buyFundFee + _sellFundFee)) / totalShare (UP168.sol#771-772)
        - amountOfLapan = toFundAmt * 12 / 13 (UP168.sol#781)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
UP168 (UP168.sol#151-1029) has incorrect ERC20 function interface:IERC20.approve(address,uint256) (UP168.sol#25)
UP168 (UP168.sol#151-1029) has incorrect ERC20 function interface:IERC20.transferFrom(address,address,uint256) (UP168.sol#27)
UP168 (UP168.sol#151-1029) has incorrect ERC20 function interface:UP168.approve(address,uint256) (UP168.sol#362-368)
UP168 (UP168.sol#151-1029) has incorrect ERC20 function interface:UP168.transferFrom(address,address,uint256) (UP168.sol#370-382)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
Reentrancy in UP168._dealBurn() (UP168.sol#495-505):
        External calls:
        - ISwapPair(_mainPair).sync() (UP168.sol#503)
        State variables written after the call(s):
        - _toBurn = 0 (UP168.sol#504)
        UP168._toBurn (UP168.sol#507) can be used in cross function reentrancies:
        - UP168._dealBurn() (UP168.sol#495-505)
        - UP168._tokenTransfer(address,address,uint256,bool,bool,bool,bool) (UP168.sol#649-724)
        - UP168._transfer(address,address,uint256) (UP168.sol#513-637)
Reentrancy in UP168._tokenTransfer(address,address,uint256,bool,bool,bool,bool) (UP168.sol#649-724):
        External calls:
        - _dealBurn() (UP168.sol#666)
                - ISwapPair(_mainPair).sync() (UP168.sol#503)
        State variables written after the call(s):
        - _takeTransfer(sender,address(this),swapAmount) (UP168.sol#676)
                - _balances[to] = _balances[to] + tAmount (UP168.sol#833)
        UP168._balances (UP168.sol#152) can be used in cross function reentrancies:
        - UP168._basicTransfer(address,address,uint256) (UP168.sol#411-420)
        - UP168._dealBurn() (UP168.sol#495-505)
        - UP168._takeTransfer(address,address,uint256) (UP168.sol#828-835)
        - UP168._tokenTransfer(address,address,uint256,bool,bool,bool,bool) (UP168.sol#649-724)
        - UP168.balanceOf(address) (UP168.sol#340-345)
        - UP168.constructor() (UP168.sol#220-319)
        - _takeTransfer(sender,address(0xdead),burnAmount) (UP168.sol#689)
                - _balances[to] = _balances[to] + tAmount (UP168.sol#833)
        UP168._balances (UP168.sol#152) can be used in cross function reentrancies:
        - UP168._basicTransfer(address,address,uint256) (UP168.sol#411-420)
        - UP168._dealBurn() (UP168.sol#495-505)
        - UP168._takeTransfer(address,address,uint256) (UP168.sol#828-835)
        - UP168._tokenTransfer(address,address,uint256,bool,bool,bool,bool) (UP168.sol#649-724)
        - UP168.balanceOf(address) (UP168.sol#340-345)
        - UP168.constructor() (UP168.sol#220-319)
        - _takeTransfer(sender,address(this),addLiquidityFeeAmount) (UP168.sol#700)
                - _balances[to] = _balances[to] + tAmount (UP168.sol#833)
```

```
INFO:Detectors:
UP168.allowance(address,address).owner (UP168.sol#356) shadows:
        - Ownable.owner() (UP168.sol#108-110) (function)
UP168._approve(address,address,uint256).owner (UP168.sol#384) shadows:
        - Ownable.owner() (UP168.sol#108-110) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
UP168.setFundAddress(address) (UP168.sol#837-841) should emit an event for:
        - fundAddress = addr (UP168.sol#839)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control
INFO:Detectors:
UP168.completeCustoms(uint256[]) (UP168.sol#874-894) should emit an event for:
        - _buyFundFee = customs[0] (UP168.sol#876)
        - _buyLPFee = customs[1] (UP168.sol#877)
        - _buyRewardFee = customs[2] (UP168.sol#878)
        - _sellFundFee = customs[4] (UP168.sol#881)
        - _sellLPFee = customs[5] (UP168.sol#882)
        - _sellRewardFee = customs[6] (UP168.sol#883)
UP168.setProcessRewardWaitBlock(uint256) (UP168.sol#969-971) should emit an event for:
        - processRewardWaitBlock = newValue (UP168.sol#970)
UP168.setHolderRewardCondition(uint256) (UP168.sol#1021-1023) should emit an event for:
        - holderRewardCondition = amount (UP168.sol#1022)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
UP168.multiAddHolder(address[]) (UP168.sol#956-962) has external calls inside a loop: ISwapPair(_mainPair).balanceOf(accounts[i]) > 0 (UP168.sol#958)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in UP168._transfer(address,address,uint256) (UP168.sol#513-637):
        External calls:
        - swapTokenForFund(numTokensSellToFund,swapFee) (UP168.sol#604)
                - _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount - lpAmount,0,toCurrencyPath,address(_tokenDistributor),block.timestamp) (UP168.sol#751-763)
                - _c.transferFrom(address(_tokenDistributor),address(this),newBal) (UP168.sol#767)
                - IWBNB(currency).withdraw(toFundAmt) (UP168.sol#777)
                - _c.transfer(fundAddress,amountOfFund) (UP168.sol#782)
                - _c.transfer(lapanAddress,amountOfLapan) (UP168.sol#783)
                - _swapRouter.addLiquidity(address(this),address(currency),lpAmount,lpCurrency,0,0,fundAddress,block.timestamp) (UP168.sol#790-803)
                - _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(_c.balanceOf(address(this)),0,rewardPath,address(this),block.timestamp) (UP168.sol#810-822)
        External calls sending eth:
        - swapTokenForFund(numTokensSellToFund,swapFee) (UP168.sol#604)
                - fundAddress.transfer(toFundAmt) (UP168.sol#778)
        State variables written after the call(s):
        - _toBurn = amount (UP168.sol#613)
Reentrancy in UP168._transfer(address,address,uint256) (UP168.sol#513-637):
        External calls:
        - swapTokenForFund(numTokensSellToFund,swapFee) (UP168.sol#604)
                - _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount - lpAmount,0,toCurrencyPath,address(_tokenDistributor),block.timestamp) (UP168.sol#751-763)
                - _c.transferFrom(address(_tokenDistributor),address(this),newBal) (UP168.sol#767)
                - IWBNB(currency).withdraw(toFundAmt) (UP168.sol#777)
                - _c.transfer(fundAddress,amountOfFund) (UP168.sol#782)
                - _c.transfer(lapanAddress,amountOfLapan) (UP168.sol#783)
```

```
INFO:Detectors:
Variable ISwapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (UP168.sol#61) is too similar to ISwapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (UP168.sol#62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
UP168.constructor() (UP168.sol#220-319) uses literals with too many digits:
        - _tTotal = 500000000000000000000000000 (UP168.sol#224)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
UP168._decimals (UP168.sol#162) should be immutable
UP168._mainPair (UP168.sol#202) should be immutable
UP168._swapRouter (UP168.sol#172) should be immutable
UP168._tTotal (UP168.sol#170) should be immutable
UP168._tokenDistributor (UP168.sol#180) should be immutable
UP168.currency (UP168.sol#173) should be immutable
UP168.currencyIsEth (UP168.sol#196) should be immutable
UP168.enableKillBlock (UP168.sol#211) should be immutable
UP168.enableOffTrade (UP168.sol#210) should be immutable
UP168.enableRewardList (UP168.sol#212) should be immutable
UP168.lapanAddress (UP168.sol#157) should be immutable
UP168.lapanAddressForToken (UP168.sol#158) should be immutable
UP168.rewardToken (UP168.sol#198) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:UP168.sol analyzed (8 contracts with 93 detectors), 83 result(s) found
```

Functional Tests
Router (PCS V2):

1- Approve (passed):
https://testnet.bscscan.com/tx/0xee88bb95a69b36de512401f6c247e15c49c417bfd5979e629ddd18d281ed3523

2- launch (passed):
https://testnet.bscscan.com/tx/0xda22bdaa056b7000eafc8bbe98c9d418b6d214241177a86350e87d651cf61cca

3- Multi Add Holder (passed):
https://testnet.bscscan.com/tx/0x1a3ed1bf5254e3d888df57bfdad106642935967c13f98a53b0064d9827da6e5c

4- Set Fund Address (passed):
https://testnet.bscscan.com/tx/0x98d624809d81273be35cc5c82402d9cbedac47c1028016e299d8e0f9691de3ff

5- Multi_bclist (passed):
https://testnet.bscscan.com/tx/0x5818c6386663aabaecb5426f7befc2164603de1be9dcbc34b7ccd9edcba89750

6- Set Add Liquidity Fee (passed):
https://testnet.bscscan.com/tx/0x7fb38df712229aed61d8254a357eb82a15ff7776676feb835898c146c1deae7e

7- Transfer (passed):
https://testnet.bscscan.com/tx/0xef7bc731c4a6cbea982495cea74de4465298da55753951c697fd92f54cc0c7c8

Ownership Privileges:

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can start trading.
- The owner can set the token sell rate.
- The owner can set an Add Liquidity Fee of not more than 25%.
- The owner can remove the liquidity fee.
- The owner can set a Fund Address.
- The owner can start/stop LP.
- The owner can set a whitelist address.
- The owner can complete customs.
- The owner can blacklist multiple addresses.
- The owner can disable the wallet limit.
- The owner can change the wallet limit.
- The owner can hold multiple addresses.
- The owner can exclude address.

Findings:
Critical: 0
High: 2
Medium: 1
Low: 3
Informational & Optimizations: 1

Centralization – Enabling Trades

Severity: High
Function: Launch
Status: Open
Overview:
The Launch function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

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

Suggestion:
To reduce centralization and potential manipulation, consider one of the following approaches:

1. Automatically enable trading after a specified condition, such as the completion of a presale, is met.
2. If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

Centralization – Owner can blacklist wallets.

Severity: High
Function: addToBlacklis
Status: Open
Overview:
The owner can blacklist wallets from transferring of tokens for an indefinite period of time which is not recommended. Which can lock user's token.

```
function multi_bclist(
        address[] calldata addresses,
        bool value
    ) public onlyOwner {
        require(enableRewardList, "disabled");
        require(addresses.length < 201);
        for (uint256 i; i < addresses.length; ++i) {
            _rewardList[addresses[i]] = value;
        }
    }
```

Suggestion:
There should be a locking period so that the wallet cannot be locked for an indefinite.
Period of time.

Centralization – Liquidity is added to EOA.

Severity: Medium
Function: addLiquidity
Status: Open
Overview:
Liquidity is added to EOA. It may be drained by the fundAddress.

```
if (lpAmount > 0 && lpCurrency > 0) {
            try
            _swapRouter.addLiquidity(
                address(this),
                address(currency),
                lpAmount,
                lpCurrency,
                0,
                0,
                fundAddress,
                block.timestamp
            )
```

Suggestion:
It is suggested that the address should be a contract address or a dead address.

Centralization – Missing Events

Severity: Low
Subject: Missing Events
Status: Open
Overview:
They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function completeCustoms(uint256[] calldata customs) external onlyOwner {
        require(enableChangeTax, "disabled");
        _buyFundFee = customs[0];
        _buyLPFee = customs[1];
        _buyRewardFee = customs[2];
        buy_burnFee = customs[3];

        _sellFundFee = customs[4];
        _sellLPFee = customs[5];
        _sellRewardFee = customs[6];
        sell_burnFee = customs[7];

        require(
            _buyRewardFee + _buyLPFee + _buyFundFee + buy_burnFee < 2500,
            "buy!<25"
        );
        require(
            _sellRewardFee + _sellLPFee + _sellFundFee + sell_burnFee < 2500,
            "sell!<25"
        );
    }
function setFundAddress(address payable addr) external onlyOwner {
        require(!isContract(addr), "no contract ");
        fundAddress = addr;
        _feeWhiteList[addr] = true;
    }
function setProcessRewardWaitBlock(uint256 newValue) external onlyOwner {
        processRewardWaitBlock = newValue;
    }
function setHolderRewardCondition(uint256 amount) external onlyOwner {
        holderRewardCondition = amount;
    }
function setisMaxEatExempt(address holder, bool exempt) external onlyOwner {
        isMaxEatExempt[holder] = exempt;
    }
function setAirdropNumbs(uint256 newValue) external onlyOwner {
        require(newValue <= 5, "!<= 5");
        airdropNumbs = newValue;
    }
```

Suggestion:
Emit an event for critical changes.

Centralization – Missing Zero Address

Severity: Low
Subject: Zero Check
Status: Open
Overview:
Functions can take a zero address as a parameter (0x00000...). If a function parameter of address type is not properly validated by checking for zero addresses, there could be serious consequences for the contract's functionality.

```
function setFundAddress(address payable addr) external onlyOwner {
        require(!isContract(addr), "no contract ");
        fundAddress = addr;
        _feeWhiteList[addr] = true;
    }
function setisMaxEatExempt(address holder, bool exempt) external onlyOwner {
        isMaxEatExempt[holder] = exempt;
    }
```

Centralization – Local variable Shadowing

Severity: Low
Subject: Variable Shadowing
Status: Open
Overview:

```
function allowance(
        address owner,
        address spender
    ) public view override returns (uint256) {
        return _allowances[owner][spender];
    }
function _approve(address owner, address spender, uint256 amount) private {
        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }
```

Suggestion:
Rename the local variables that shadow another component.

Optimization

Severity: **Informational**

Subject: Floating Pragma Solidity version

Status: Open

Overview:

It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

```
pragma solidity ^0.8.18;
```

Suggestion:

Adding the latest constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.