Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

UP611

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	2
Medium	1
Low	3
Informational	1

Centralization Risks

Owner Privileges	Description
Can Owner Set Taxes >25% ?	Not Detected
Owner needs to enable trading ?	Detected
Can Owner Disable Trades ?	Not Detected
Can Owner Mint ?	Not Detected
Can Owner Blacklist ?	Detected
Can Owner set Max Wallet amount ?	Not Detected
Can Owner Set Max TX amount ?	Not Detected

TABLE OF CONTENTS

OVERVIEW

The Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result	**High risk Major Flag**
Audit Date	**14 June 2024**

CONTRACT DETAILS

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

AUDIT METHODOLOGY

Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch , that lead to scams and rugpulls.

Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability

Tools

- Manual Review: The code has undergone a line-by-line review by the Ace team.
- 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.
- Slither: The code has undergone static analysis using Slither.

VULNERABILITY CHECKS

Design Logic	Passed
Compiler warnings	Passed
Private user data leaks	Passed
Timestamps dependence	Passed
Integer overflow and underflow	Passed
Race conditions & reentrancy. Cross-function race conditions	Passed
Possible delays in data delivery	Passed
Oracle calls	Passed
Front Running	Passed
DoS with Revert	Passed
DoS with block gas limit	Passed
Methods execution permissions	Passed
Economy model	Passed
Impact of the exchange rate on the logic	Passed
Malicious event log	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross-function race conditions	Passed
Safe Zepplin module	Passed

RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and acces control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

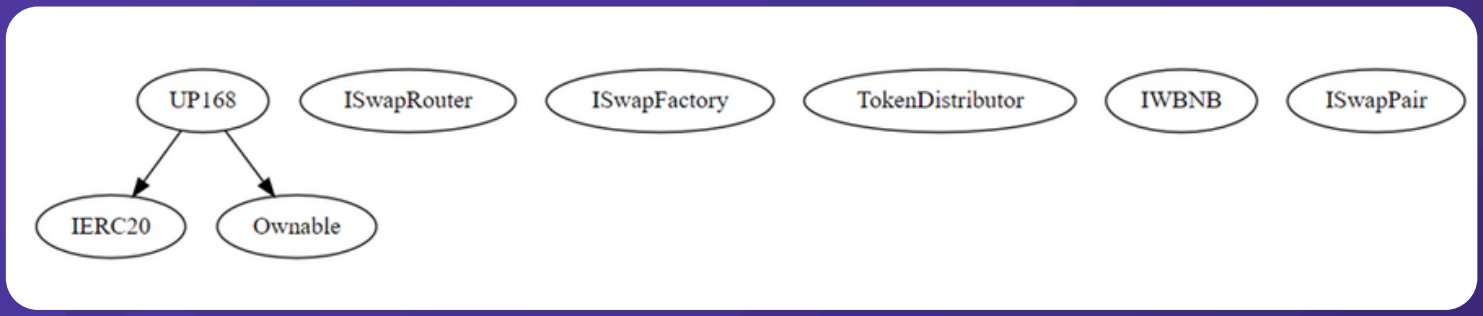
Low Risk

Issues on this level are minor details and warning that can remain unfixed.

Informational

Issues on this level are minor details and warning that can remain unfixed.

INHERITANCE TREE



expelee

STATIC ANALYSIS

```
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)
```

expelee

STATIC ANALYSIS

```
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
```

TESTNET VERSION

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

MANUAL REVIEW

Severity Criteria

Expelee assesses the severity of disclosed vulnerabilities according to methodology based on OWASP standarts.

Vulnerabilities are dividend into three primary risk categroies:
High
Medium
Low

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious input handling
- Escalation of privileges
- Arithmetic
- Gas use

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

HIGH RISK FINDING

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.

HIGH RISK FINDING

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.

MEDIUM RISK FINDING

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.

LOW RISK FINDING

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;
  }
```

LOW RISK FINDING

```
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.

LOW RISK FINDING

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;
  }
```

LOW RISK FINDING

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.

INFORMATIONAL & OPTIMIZATIONS

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.

ABOUT EXPELEE

Expelee is a product-based aspirational Web3 start-up. Coping up with numerous solutions for blockchain security and constructing a Web3 ecosystem from deal making platform to developer hosting open platform, while also developing our own commercial and sustainable blockchain.

www.expelee.com

expeleeofficial
expelee
Expelee
expelee
expelee_official
expelee-co

expelee

Building the Futuristic Blockchain Ecosystem

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 guarantess against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always do your own research and project yourselves from being scammed. The Expelee 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 Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Alway 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 Expelee team disclaims any liability for the resulting losses.