Submitted audit for **Jungle** at 18 june 2023
Audit result : **Passed**

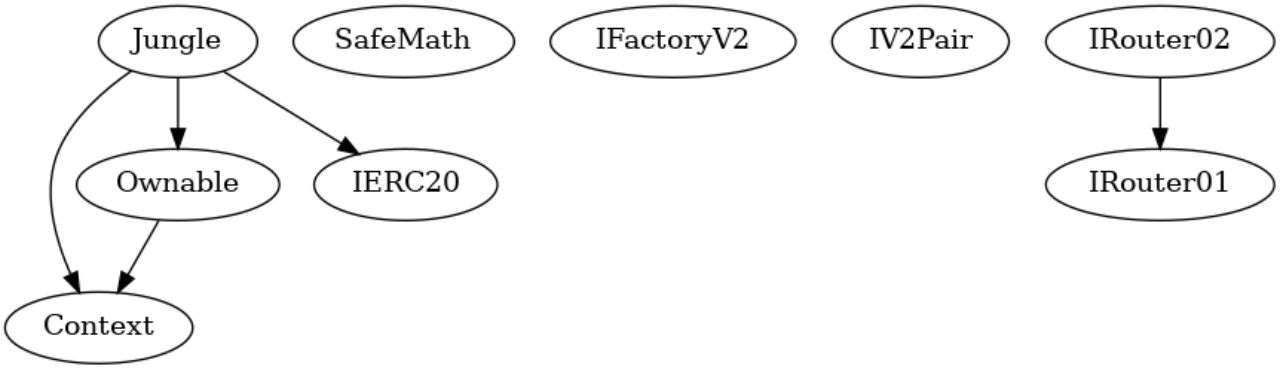
Token Address: 0xb8d5C50f2FDc740D06DF92f0F5194b1a994b7Fe9
Name: Jungle
Symbol: $JUNGLE
Decimals: 18
Netowrk: Binance smart chain
Token Type: BEP20
Owner: 0x368ffBD1e34EAB57F8bCe0F94F47F02807019d77
Deployer: 0x368ffBD1e34EAB57F8bCe0F94F47F02807019d77
Token Supply: 100,000,000,000
Checksum:
2150c99ae7d31b0a27ee43246ba55b567a889669
Testnet version:
The tests conducted were performed on the contract deployed on the Binance Smart Chain (BSC) Testnet.
https://testnet.bscscan.com/address/0x9246087965CD732021C7575C2617C9e536f6622F#code

The following tools were utilized to ensure the reliability and functionality of the code:

1. Manual Review: The code underwent a thorough review process by the Expelee team, which involved analyzing the code line by line. This review helped to identify any potential issues or bugs that could impact the performance of the code.
2. BSC Test Network: The code was tested extensively on the BSC Test network to ensure that it functions as intended. All tests were conducted using the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be accessed in the "Functional Tests" section of the report.

Overall, these tools were instrumental in identifying any potential issues and ensuring that the code functions as expected.

Inheritance:



Contract Assesment

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation				
└	<Constructor>	Public ❗	🛑	NO ❗
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context			
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_setOwner	Private 🔐	🛑	
SafeMath	Library				
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
IFactoryV2	Interface				
└	getPair	External ❗		NO ❗
└	createPair	External ❗	🛑	NO ❗
IV2Pair	Interface				
└	factory	External ❗		NO ❗
└	getReserves	External ❗		NO ❗
└	sync	External ❗	🛑	NO ❗
IRouter01	Interface				
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	addLiquidity	External ❗	🛑	NO ❗
└	swapExactETHForTokens	External ❗	💵	NO ❗
└	getAmountsOut	External ❗		NO ❗
└	getAmountsIn	External ❗		NO ❗
IRouter02	Interface	IRouter01			
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗

└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForTokens	External ❗	🛑	NO ❗
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	decimals	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	name	External ❗		NO ❗
└	getOwner	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
Jungle	Implementation	Context, Ownable, IERC20		
└	<Receive Ether>	External ❗	💵	NO ❗
└	<Constructor>	Public ❗	🛑	NO ❗
└	totalSupply	External ❗		NO ❗
└	decimals	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	name	External ❗		NO ❗
└	getOwner	External ❗		NO ❗
└	allowance	External ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	transferFrom	External ❗	🛑	NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	_transfer	Internal 🔒	🛑	
└	checkCommunityWallet	Internal 🔒	🛑	inTxWinnerFlag
└	internalSwap	Public ❗	🛑	NO ❗
└	swapTokensForEth	Internal 🔒	🛑	inSwapFlag
└	takeTaxes	Internal 🔒	🛑	
└	approve	External ❗	🛑	NO ❗
└	_approve	Internal 🔒	🛑	
└	isBuy	Internal 🔒		
└	isSell	Internal 🔒		
└	isTransfer	Internal 🔒		
└	addLpPair	External ❗	🛑	onlyOwner
└	isExcludedFromFee	External ❗		NO ❗
└	increaseCommunityPrize	Public ❗	🛑	NO ❗
└	setIsExcludedFromFee	Public ❗	🛑	onlyOwner
└	setMarketingFee	Public ❗	🛑	onlyOwner
└	setCommunityFee	Public ❗	🛑	onlyOwner
└	setCommunityMinimumSeconds	Public ❗	🛑	onlyOwner
└	setSwapThreshold	Public ❗	🛑	onlyOwner
└	getCommunityInfo	Public ❗		NO ❗

└	getLastCommunityWinner	Public ❗		NO ❗
└	rescueStuckBEP20	Public ❗	🛑	onlyOwner
└	rescueStuckETH	Public ❗	🛑	onlyOwner
└	getTotalBurned	Public ❗		NO ❗
└	enableTrading	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Testnet Version:

Adding Liquidity ✅
Tx:
https://testnet.bscscan.com/tx/0x4499b99644ee54c258c082462f25051664d146b106f6cc8949f1fb96771d1008

==========

Buying from a fee excluded wallet ✅
Tx (0% tax):
https://testnet.bscscan.com/tx/0x4e1fe63dac43c000660ba836a3bea5ea55dd04063adfaa4a2e69c84e49229694

==========

Selling from a fee excluded wallet ✅
Tx (0% tax):
https://testnet.bscscan.com/tx/0x3f16790eea7ec16f7ffd6fb303a5b563116d38385122bf7c946f506119fe32c3

==========

Transferring using a fee excluded wallet ✅
Tx (0% tax):
https://testnet.bscscan.com/tx/0x692a89ad8d9d0fc46533190886c4a973211ad24c54f2222b8181d53ed2131902

==========

Buying from a regular wallet ✅
Tx (0-8% tax):
https://testnet.bscscan.com/tx/0xa98a8f306f2e86577fb47eca053b8ac7d599dbafcaeabb27f49010d70993cb56

==========

Selling from a regular wallet ✅
Tx (0-8% tax):
https://testnet.bscscan.com/tx/0x5bd6a96ff8cb3e23a91a02cfdc47b7a37ec3191cf677ca4372527c1147fc5b79

==========

Transferring from a regular wallet ✅
Tx (0-8%):
https://testnet.bscscan.com/tx/0xf4d9566311d6d554b006d960e1335a960c6e6eb0d46c465cfaf3f5ebadd9020d

==========

Internal swap (Marketing BNB) ✅
Tx:
https://testnet.bscscan.com/address/0x368ffbd1e34eab57f8bce0f94f47f02807019d77#tokentxns

==========

Key points:

- Contract owner is not able to set up to 8% tax for buy/sell/transfers
- Contract owner can not mint new tokens
- Contract owner can not disable trades
- Contract owner can not blacklist wallets
- Contract owner is not able to set max buy/sell/transfer
- Contract owner must enable trades manually

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

Category: Logical
Subject: Invalid balance modifications
Status: Open

Severity: Critical

Overview:

The method takeTaxes is returning a uint256 value which denotes the transfer amount post deductions from fees. However, an issue arises within the _transfer function where the amount post tax deduction is inaccurately deducted from the token sender's balance, while the original, unaltered transfer amount is added to the recipient's balance. This inconsistency leads to incorrect balance calculations.

```
uint256 amountAfterTaxes = takeTaxes(from, to, amount);
balance[from] = balance[from].sub(amountAfterTaxes, "Insufficient Balance");
balance[to] = balance[to].add(amount);
```

Suggestion:

Modify balance correctly:

```
uint256 amountAfterTaxes = takeTaxes(from, to, amount);
balance[from] = balance[from].sub(amount, "Insufficient Balance");
balance[to] = balance[to].add(amountAfterTaxes);
```

In the revised code, the total transfer amount (including tax) is subtracted from the sender's balance (balance[from]), and only the amount after tax deductions (amountAfterTaxes) is added to the recipient's balance (balance[to]). This modification ensures an accurate balance calculation.

Category: Denial of Service (DoS)

Subject: Potential Denial of Service (DoS) through Withdrawal of Native Tokens Status: Open

Status: Open
Impact: Critical
Overview:
The contract owner currently has the ability to withdraw any ERC20 token from the contract. If the native tokens (Jungle Token) are withdrawn, there will not be sufficient tokens (communityAccumulatedTokens) left in the contract to transfer to the winner. This can potentially disrupt the normal functioning of the contract.

An extreme scenario arises when communityMinimumSeconds has elapsed. In such a case, if the contract contains zero tokens and attempts to transfer communityAccumulatedTokens to a winner (irrespective of whether the caller of the contract is the owner or a whitelisted participant), all contract operations would be indefinitely suspended.

```
  function rescueStuckBEP20(address tokenAddres, address to, uint256
amount) public onlyOwner {
      IERC20(tokenAddres).transfer(to, amount);
  }
```

Suggestion:
Ensure that owner is not able to withdraw jungle tokens from the contract

```
  function rescueStuckBEP20(address tokenAddres, address to, uint256
amount) public onlyOwner {
      require(tokenAddress!=address(this), “Can’t withdraw jungle tokens”);
      IERC20(tokenAddres).transfer(to, amount);
  }
```

Category: Denial of Service (DoS)
Subject: Potential Denial of Service (DoS) if communityAccumulatedTokens is 0
Status: Open
Impact: Critical
Overview:
If a winner is choosen (I.e communityMinimumSeconds has elapsed) but communityAccumulatedTokens is set to 0, all the actions in the contract will be disabled to the point that even owner of the contract or wallets that are excluded from fees wont be able to do any actions (buy/sell/transer). This is because transferring communityAccumulatedTokens to winner is done before increasing communityAccumulatedTokens by receiving fees.

```
function checkCommunityWallet(address from, address to, uint256 amount) internal
inTxWinnerFlag returns (bool) {
   // Some code...
   _transfer(address(this), communityWinnerWallet, communityWinnerTokens);
   // Some code...
}
```

Suggestion:
The function checkCommunityWallet should exit if communityAccumulatedTokens equals 0, or if the contract's balance is less than communityAccumulatedTokens. This change will prevent a potential Denial of Service scenario.

Here's the revised function:

```
function checkCommunityWallet(address from, address to, uint256 amount) internal inTxWinnerFlag returns (bool) {
   // if communityFee == 0 means community wallet is disabled
   if (communityFee == 0) {
      return false;
   }

 if (communityWalletIsEmpty == false && block.timestamp >  (communityBuyTimestamp +
communityMinimumSeconds)) {
      // if communityAccumulatedTokens is 0, exit function
      if (communityAccumulatedTokens == 0){
            return false;
      }
      // if contract balance is less than communityAccumulatedTokens, exit function
      if(balanceOf(address(this)) < communityAccumulatedTokens){
           return false;
      }
       // Remaining code…
 }
}
```

These changes ensure that if there are insufficient tokens in the contract or the accumulated tokens are zero, the contract will not attempt to perform transfers that could lead to a halt in its operation.

Category: **Centralization**
Subject: Enabling trades is not guaranteed
Status: Open
Impact: **High**
Overview:
Owner must enable trades for investors manually. If trades remain disabled, holders wont be able to trade their tokens.

```
function enableTrading() external onlyOwner {
    require(!isTradingEnabled, "Trading already enabled");
    isTradingEnabled = true;
    emit TradingEnabled();
}
```

Suggestion:
There are multiple ways to resolve this issue:

- Enable trades prior to launch or presale : this ensurs that trades are already enabled forver and can not be disabled later. However, this completely negates the need for enableTrading function.
- Transfer Ownership of the contract to a trusted 3rd party: You can transfer ownership of the contract to a trusted 3rd party e.g a certified pinksale safu developer

Create a time lock contract for enabling trades: this contract can enable trades after a fixed period of time.

Category: Logical
Subject: communityWallet is not reset after selling
Status: Open
Impact: Medium
Overview:
if communityWallet sold/transfer their tokens, communityWallet is not reseted to address(0). Which means communityWallet is still eligible for receiving communityAccumulatedTokens.

```
    // current community wallet is involved in tx
    if (from == communityWallet || to == communityWallet) {
        bool timeCompleted = block.timestamp > (communityBuyTimestamp +
communityMinimumSeconds);

        // if isn't a buy wallet is disqualified, set max buy to 0
        if (isBuy(from, to) == false && timeCompleted == false) {
            communityWalletIsEmpty = true;
            communityBuyEthAmount = 0;
            return false;
        }
    }
```

Suggestion:
first, ensure that communityWallet is reseted after selling/transferring their tokens:

```
    // current community wallet is involved in tx
    if (from == communityWallet || to == communityWallet) {
        bool timeCompleted = block.timestamp > (communityBuyTimestamp +
communityMinimumSeconds);

        // if isn't a buy wallet is disqualified, set max buy to 0
        if (isBuy(from, to) == false && timeCompleted == false) {
            communityWalletIsEmpty = true;
            communityBuyEthAmount = 0;
            communityWallet = address(0);
            return false;
        }
    }
```