Contract: PreSale Contract

Audited by Muhammad Talha

These are the following bugs, issues and minor changes need to be fixed in tis contracts

Total : 33

```
contract Presale {

    using SafeMath for uint256;

    ERC20 token; //define visibility

    address public wallet;
    address public owner;
```

//audit :- define the visibility of the variable (ERC20 token).

```
receive() external payable { //audit: remove it there is no need of fallback function.
    buyTokens(msg.sender);
}
```

This is an extra call back function. There is no need for a call back function in this contract.

```
function startPresale(uint256 _rate) public { // audit:
```

//audit: Make the visibility of this function external because this gonna be call by outside of the contract only.

```
//audit:  this require statement use almost all the function best practice is to make a modifier.
require(msg.sender == owner, "Only Owner can start PreSale");
```

//audit: this requires statement use almost all the function best practice is to make a modifier.

```
require(isPreSaleStarted == false, "Presale is already started");
require(_rate > 0); // give the proper message to understand the error
```

//audit : The second require statement gives a proper message to understand the error like all other require statements in the whole contract.

//audit: Make the visibility of this function external because this gonna be call by outside of the contract only.

```
function changeRate(uint256 _rate) public { // audit: make the visibility of this function external

    require(msg.sender == owner, "Only Owner can change Rate"); // audit: use the modifier instead.
    require(isPreSaleStarted == true,"You can only change rate after presale is started");
```

//audit: use the onlyOwner modifier instead of the first required statement.

```
function changeRate(uint256 _rate) public { // audit: make the visibility of this function external

    require(msg.sender == owner, "Only Owner can change Rate"); // audit: use the modifier instead.
    require(isPreSaleStarted == true,"You can only change rate after presale is started");

    require(_rate > 0); // give the proper message to understand the error
```

//audit: Third, require statements give a proper message to understand the error like all other require statements in the whole contract.

```
function changeRate(uint256 _rate) public { // audit: make the visibility of this function external

    require(msg.sender == owner, "Only Owner can change Rate"); // audit: use the modifier instead.
    require(isPreSaleStarted == true,"You can only change rate after presale is started");

    require(_rate > 0); // give the proper message to understand the error
    emit rateChanged(rate, _rate); //move to the end of the function
```

//audit: the emit statement must be at the end of the function like all other functions and it is kinda good practice to put all the event statements at the end of the functions or methods.

```
function whitelistSender(address sender) public { // audit:
```

//audit: Make the visibility of this function external because this gonna be call by outside of the contract only.

```
function whitelistSender(address sender) public { // audit: make the visibility of this function external it o

    require(msg.sender == owner, "Only Owner can add Whitelist address"); // audit: use the modifier instead.
```

//audit: use the onlyOwner modifier instead of the first required statement.

```
    function whitelistSender(address sender) public { // audit: make the visibility of this function external it onl

        require(msg.sender == owner, "Only Owner can add Whitelist address"); // audit: use the modifier instead.
        // audit: add a require statement to check that the sender address wouldent be zero.
        senderWhitelist[sender] = true;
        useSenderWhitelist = true; //didnt need this variable
    }
```

//audit: add a required statement to check that the sender address wouldn't be zero.

```
function blacklistSender(address sender) public {  // audit:
```

//audit: Make the visibility of this function external because this gonna be call by outside of the contract only.

```
function blacklistSender(address sender) public {  // audit: make the visibility of this function external
    require(msg.sender == owner, "Only Owner can blacklist address"); // audit: use the modifier instead.
```

//audit: use the onlyOwner modifier instead of the first require statement.

```
function blacklistSender(address sender) public {  // audit: make the visibility of this function external it or
    require(msg.sender == owner, "Only Owner can blacklist address"); // audit: use the modifier instead.
    // audit: add a require statement to check that the sender address wouldent be zero.
    senderWhitelist[sender] = false;
}
```

//audit: add a required statement to check that the sender address wouldn't be zero.

```
function changeWalletAddress(address newWallet) public { // audit:
```

//audit: Make the visibility of this function external because this gonna be call by outside of the contract only.

```
function changeWalletAddress(address newWallet) public { // audit: make the visibility of this function exter

 require(msg.sender == owner, "Only Owner can change Wallet address"); // audit: use the modifier instead.
 wallet = newWallet;
}
```

//audit: use the onlyOwner modifier instead of the first require statement.

```
function changeWalletAddress(address newWallet) public { // audit: make the visibility of this function external

 require(msg.sender == owner, "Only Owner can change Wallet address"); // audit: use the modifier instead.

 //audit: add a require statement to check that the wallet address wouldent be zero.
 wallet = newWallet;
}
```

//audit: add a required statement to check that the wallet address wouldn't be zero.

```
function buyTokens(address _beneficiary) public payable {  // audit:

    require(isPreSaleStarted == true, "Presale is not started yet");
```

//audit: Make the visibility of this function external because this gonna be call by outside of the contract only.

```
function buyTokens(address _beneficiary) public payable {  // audit: make the visibility of this function external it only ca

    require(isPreSaleStarted == true, "Presale is not started yet");



    if (useSenderWhitelist) { // donot need this if statement kindly remove this.
        require(senderWhitelist[msg.sender], "sender not whitelisted");
    }

    // audit : ity would be checked first thta the message value != 0.
    uint256 weiAmount = msg.value;
    _preValidatePurchase(_beneficiary, weiAmount); // audit: no need an extra function just put those require statement here.
```

//audit: there must be a require statement to check the msg.value wouldn't be zero before storing it into the well Amount.

//audit: remove an extra _prevalidatePurchase function call and add those two simple require statements in this function to save the extra gas cost.

```
    // calculate token amount to be created
    uint256 tokens = _getTokenAmount(weiAmount);

    // update state
    weiRaised = weiRaised.add(weiAmount);
    // audit : evternal call must ne happened at the end of the function.
    _processPurchase(_beneficiary, tokens);// audit: remove this function
```

//audit: remove this extra function call _processPurchase instead send the token here directly in this function this would save a lot of gas fee.

//audit: this send tokens call must be the end of the function for contract safety.

```
uint256 tokens = _getTokenAmount(weiAmount);

// update state
weiRaised = weiRaised.add(weiAmount);
// audit : evternal call must ne happened at the end of the function.
_processPurchase(_beneficiary, tokens);// audit: remove this function call thsi function isnt doing any thing.
emit TokenPurchase(msg.sender, _beneficiary, weiAmount, tokens); //move to the end of the function
```

//audit: the emit statement must be at the end of the function like all other functions and it is kinda good practice to put all the event statements at the end of the functions or methods.

//audit: the getTokenAmount function is not right, it isnt giving the right amount of tokens.

```
_updatePurchasingState(_beneficiary, weiAmount);//audit: remove this function this function isnt doing anythiog.
```

//audit: this is an extra function call in the function which isn't doing anything, so remove it.

```
  _postValidatePurchase(_beneficiary, weiAmount);//audit: remove this function
```

//audit: this is an extra function call in the function which isn't doing anything, so remove it.

```
// audit : there is no need of this function remove it and use the require statements in same functions.
function _preValidatePurchase(address _beneficiary, uint256 _weiAmount) internal pure{
```

//audit: this is an extra function to remove it and add those require statement in the same function to prevent it from usage of extra gas fee.

```
    require(_beneficiary != address(0)); // audit: give proper message for the error
```

//audit: In this require statement give a proper message to understand the error like all other required statements in the whole contract.

```
    require(_weiAmount != 0); // audit : give proper message message  for the error
```

//audit: In this require statement give a proper message to understand the error like all other required statements in the whole contract.

```
  // audit this function isnt doing anything kindly remove it.
function _postValidatePurchase(address _beneficiary, uint256 _weiAmount)
    internal
{
    // optional override
}
```

//audit: remove this extra function which isn't doing anything in the contract.

```
// audit : there is no need of this function remove it and transfer token in the same function.
function _deliverTokens(address _beneficiary, uint256 _tokenAmount)
    internal
```

//audit: remove this extra function and send those tokens in the same function to prevent the extra gas cost.

```
    internal
{
    token.transfer(_beneficiary, _tokenAmount);// this must be in require statement to check if token send or not.
}
```

//audit: this token transfer call must be in a require statement to prevent if somehow token wouldn't send we be able to check and send the proper error message.

```
// audit : there is no need of this function remove it
function _processPurchase(address _beneficiary, uint256 _tokenAmount) internal{
    _deliverTokens(_beneficiary, _tokenAmount);
}
```

//audit: remove this function which isn't doing anything except making another function call.

```
// audit this function isnt doing anything kindly remove it.
function _updatePurchasingState(address _beneficiary, uint256 _weiAmount)
    internal
{
    // optional override
}
```

audit: remove this extra function which isn't doing anything in the contract.

```
function _forwardFunds() internal {
    payable(wallet).transfer(msg.value);
}
```

audit: make the visibility of this function private so not another contract can call this.