Audit Report

- No need to create storage variables for storing error messages. Instead, hardcode error message in a require statement, it will save deployment gas cost.
- Decimal is already initialized so no need to ask for decimal in constructor argument.
- Use owner variable instead of msg.sender in constructor after you have initialized owner variable in constructor.
- Change modifiers of functions from public to external, which are not being called inside the contract. Example, resetAllowed, setTrading, etc. Gas effective.
- For better readability, use bool instead of uint256 for "tradingAllowed" variable.
  - True = trading is allowed, false = not allowed. In addition, a function which will just toggle the variable. I.e. make it true when it's false, make it false when it's true.
- In "detectTransferRestriction" function, require on line 179 and 180 should be rewritten, with first require checking whether the seller and buyer is restricted or no, and the second require checking whether time is passed or no. Also, condition should be <= or >= instead of just < & >.

  These two require statements could be changed from:

```
-    require(  _sellRestriction[_from] != 0 && _sellRestriction[_from] < block.timestamp,
     NotWhitelisted  );
-            require(  _buyRestriction[_to] != 0 && _buyRestriction[_to] < block.timestamp,
     NotWhitelisted  );
```

to:

```
        require(  _sellRestriction[_from] != 0 && _buyRestriction[_to] != 0, "Not whitelisted"  );
        require(  _sellRestriction[_from] <= block.timestamp&& _buyRestriction[_to] <=
block.timestamp, "Time has not passed"  );
```

- There is no need to use safeMath library, as from solidity 0.8, underflow and overflow checks are enabled by default.
- Admin can burn anyone's token right now. This should not happen in my opinion because people can see the code, and they can see that admin can burn his tokens whenever admin wants, hence user will not get real sense of ownership.
- Code should compile without warnings. There are warnings of previous declarations and unused variables.

- Buy and sell restriction is same for all tokens right now, regardless of when they were minted. So if admin mints 1000 tokens at my address on 21 nov and sets the restriction to end on 25th nov and then mint another 1000 tokens at my address on 23 nov, then restriction of those 1000 tokens will also end on 25th nov. Similarly, if admin updates it to end on 28 novemeber after minting new 1000 tokens, then restriction of old 1000 tokens will also end on 28 nov. Not sure if that is intentional.
- In "detectTransferRestriction" we are checking that if there is a limit on number of investors then before transferring we check whether current investor count is less than allowed investor or not. Consider a scenario where there is a limit of 10 investors, and now I want to transfer my tokens to an address which has 0 balance of the token, I won't be able to transfer since there is a limit of 10 investors. Now, if I, sender, is transferring all token balance, then technically the investor count should be dropped to 9 and it should allow me to transfer the tokens to recipient, so if limit of investor is reached, and then the sender is trying to transfer each token balance to the recipient whose token balance is 0, he should be able to do so, which is not possible right now.
- It is always better to use openzeppelin code as their contract is battle tested. It will not increase deployment cost by much. Will strongly suggest to use open zeppelin's erc20 contract.
- Following are the issues with your implementation of erc20 code:
    1. You are not even inheriting IERC20 interface. If you are not inheriting open zeppelin's contract, at least inherit the interface.
    2. Your transfer, transferFrom and other functions are not returning bool values, which means that they cannot be integrated with many DEX, like balancer. Your erc20 functions must return true/ false.
    3. Your transfer and transfer from function have reduntant code; make a shared logic for transfer and transferFrom function to avoid reduntant code.
    4. It is prone to approve/ transferFrom bug, as stated in the document: https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM/edit
    5. transferFrom function should not subtract allowance if the owner has approved the spender for max amount of uint256.
- All of the above 5 issues will be resolved by inheriting and overriding open zeppelin's ERC 20 contract.
- Always use proper error messages, they should not be vague. For example, error message is same if I am transferring tokens from someone else address and the owner has not approved me to spend the said amount or if I am the owner myself and want to transfer tokens with more than my balance.