V1.0

WePower

Audit

Dean Eigenmann - Zero Knowledge Labs
December 19, 2017

Disclaimer **2**

Overview **2**

Audit Results **3**

Contribution.sol **4**

- Suggestions 4
- Modifiers 4
  - Initialized 4
  - contributionOpen 4
  - notPaused 4
- Constructor 4
- Initialize 4
  - Suggestions 4
- blacklistAddresses 5
- whitelistAddresses 5
- whitelist 5
- blacklist 5
- tokensToGenerate 5
- proxyPayment 5
  - Suggestions 5
- doBuy 6
  - Issues 6
  - Suggestions 6
- Finalize 6
  - Suggestions 7
- weiToCollect 7
  - Suggestions 7
- claimTokens 7
- pauseContribution 7

Exchanger.sol **8**

- Constructor 8
- Default Function 8
  - Suggestions 8
- collect 8
  - Suggestions 8
- claimTokens 9

Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bugfree status. The audit documentation is for discussion purposes only.

Overview

The audit was performed on all smart contracts found the WePower contracts repository (https://github.com/WePowerNetwork/wepower-contracts), the contracts are found in the directory: contracts. The commit hash this audit was performed upon is: 138b776d5afab9ac305e85788020325a8ceb6683

Audit Results

The code audited contains various issues that should be fixed, along with those issues a list of various suggestions has been provided that should be considered.

Contribution.sol

This contract is responsible for the main crowdsale logic, this includes purchasing tokens as well as calling the exchanger for those participants who have received presale tokens.

Suggestions

1. Use an actual constructor rather than a custom initialize function. This will prevent the requirement of tracking the initialized state manually.

Modifiers

Initialized

Ensures that the value of initializedBlock is not equal to zero.
Suggestion: Checks like these should be a require and not an assert.

contributionOpen

Ensures that the block timestamp is between the start and ending time, and that the sale has not yet been finalized.
Suggestion: Checks like these should be a require and not an assert.

notPaused

Ensures the value of paused is equal to false.

Constructor

Sets the WPR

Initialize

This function sets various passed parameters related to the crowdsale. The total supply of the WCT, WCT1 and WCT2 tokens at the time of initialization are added to the weiPreCollected variable.
The tokens are minted to the exchanger address, the amount minted is weiPreCollected times 1250. This accounts for the 25% bonus.

Suggestions

1. Merge this function with the constructor to reduce complexity.

2. The asserts on lines 78-80 and 95 should be requires.

blacklistAddresses

Calls blacklist function for an array of addresses.

whitelistAddresses

Calls whitelist function for an array of addresses.

whitelist

Increments the value of numWhitelistedInvestors and sets the value of canPurchase for the specified address to true.

blacklist

Decrements the value of numWhitelistedInvestors and sets the value of canPurchase for the specified address to false.

tokensToGenerate

It is checked if the totalWeiCollected is smaller than the bonus cap.
If this is the case, it is checked how many tokens can still be bought with the bonus. Then tokens are generated with the bonus which are still allowed. Finally toFund is subtracted by the amount of wei that had the bonus.

Finally tokens are generated for the amount that is passed in toFund, this is added to the already generated tokens incase we had an amount with tokens.

proxyPayment

This function either calls the collect function on the Exchanger if no ether was sent, otherwise it calls the doBuy function.

Suggestions

1. Unpausing and pausing the token on every purchase is inefficient, the crowdsale should still be allowed to transfer tokens even when the token is paused.
2. As the return value is never used, it can be removed.

doBuy

The function is responsible for the logic of token sale, this includes minting as well as accounting for the sold tokens.

It is ensured that if the current time has not surpassed the first sale day, that the user is in the whitelist.
It is ensured that the amount of ether sent is greater than or equal to the minimum transaction amount.
The value of the variable toFund is set to the amount of ether sent.
It is ensured that more than the cap has not been hit.
If the amount remaining till the cap is reached is smaller than the sent amount, the crowdsale sets the value of toFund to this value.
It is calculated how many tokens should be generated.
It is ensured that more than 0 tokens will be generated.
The tokens required are minted for the user.
The amount specified in the toFund variable is sent to the contribution wallet.
The mapping that tracks the amount of contributions is updated for the buyer address.
The value of toFund is added to the totalWeiCollected variable.
If the amount sent minus the value of toFund is larger than 0, then this amount is returned to the buyer.

Issues

1. This function never checks the endTime of the sale, meaning the sale can go on until the owner decides to finalize the sale or the cap is completely reached.

Suggestions

1. The if statement on line 209 checking if the cap has not been reached, should be wrapped in a require statement. This will automatically refund all the eth sent and does not require a manual transaction. This will allow removal of the else case.

Finalize

It is ensured that the crowdsale has not been finalized yet, additionally it is ensured that the current timestamp is passed the start and end time. It is then checked that the cap was reached and that the caller is the owner.
Various allocations are minted.
Token function finishMinting is called.
Ownership of the token is transferred to the caller.
Finalization timestamp and block number is set.

Suggestions

1. Instead of saving both the block number and timestamp to ensure finalization, a simple boolean can be saved called finalized which is set to true at the end of the function.
2. Add onlyOwner modifier instead of checking the owner in a require.
3. Checking that the current time is passed the end time is enough, we don't need to check it has passed the startTime.

weiToCollect

If the totalWeiCollected is greater than the totalWeiCap, zero is returned. Otherwise, totalWeiCap subtracted by totalWeiCollected is returned.

Suggestions

1. The check should not be for greater than, but rather greater than or equal to. There is no point in calculating when we know it is zero.

claimTokens

Allows owner to drain tokens or ether that was deposited into the contract. This also calls the same function on the token contract.

pauseContribution

Sets the value of paused to the passed boolean.

Exchanger.sol

Constructor

The constructor sets the 3 contribution tokens that are passed, as well as the final WePower token. Additionally the Contribution address is set.

Default Function

If the sale has not been finalized, then the contribution contract is called else the collect function is called.

Suggestions

1. This function seems out of place, and should be removed.

collect

This function allows users who have the various contribution tokens to exchange them for the final WePower token.

It is ensured that the crowdsale has started.
The initialization block of the crowdsale is stored as the presale fixed block.
The balance for the caller at the presale fixed block is investigated for the 3 tokens.
The total supply of the 3 tokens at the presale fixed block is investigated.
The balance for the final token is investigated for the caller, it is added together with the totalCollected variable.
The total is multiplied by the added up balance of the 3 tokens, and then divided by the added up total supplies.
The calculated value is subtracted by the value found in the collected mapping.
It is ensured that the calculated value is greater than zero.
The calculated value is added to the totalCollected variable.
The calculated value is added to the value in the collected mapping.
The tokens tokens are transferred to the caller, the amount being the calculated value.

Suggestions

1. Both of the asserts in this function should be changed to a require.

claimTokens

Allows owner to drain tokens or ether that was deposited into the contract.