V1.0

DataWallet

Audit

Dean Eigenmann - Zero Knowledge Labs

January 23, 2017

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 DataWalletCrowdsale core repository (https://github.com/DataWallet/crowdsale), the contracts are found in the directory: contracts. The commit hash this audit was performed upon is:
bc65beba45faf8c6bfc8065d6228c8ec26d73da4

The contract TokenVesting.sol was ignored, this is due to the fact that it is from the already audited Open Zeppelin code base.

The contracts written by the DataWallet contracts are based on contracts previously audited by ZK Labs and written by the SingularityNet team.

Audit Results

Generally speaking the code style is good and various tests have been supplied. There are minor suggestions to improve the legibility as well as the functionality of the code.

DataWalletCrowdsale.sol

This contract contains all the logic for the DataWallet crowdsale. The contract inherits from the Zeppelin Contracts Ownable and Reentrancy guard. It uses SafeMath for all uint256 variables.

Constructor

The constructor function sets the vault, token, wallet, startTime, endTime, firstDay, firstDayCap, rate, goal and cap.

Suggestions

1. Wallet can be removed as it is never used, this is due to the fact that it is set twice once being as the vault.

Default Function

The default function calls the “buyTokens” function.

buyTokens

This function is responsible for the main sale logic.

The contract ensures that the passed beneficiary is not the burn address, that the beneficiary has been whitelisted and that the purchase is valid.
If the purchase is being done during the first day, it is ensured that the beneficiary will not exceed the first day cap.
It is then checked how much eth can still be contributed, if the sent ether exceeds this amount it is calculated how much will need to be returned.
The amount that is allowed to be contributed is forwarded to the vault.
If eth needs to be returned, it is sent back to the caller of the function.
It is calculated how many tokens a user will get for a contributed amount.
The amount contributed is added to both the weiRased and the contributions for a specific address variables.
The tokens are then transferred using the transferTokens function.

Suggestions

1. Stick to either using beneficiary or msg.sender, but not both.
2. Use default ERC20 transfer function instead of custom transferTokens function.

getTokens

Multiplies tokens by the specified rate and then divides it by “WEI_TO_DATUM” turning it into an 8 decimal token.

claimRefund

If the sale has been finalized and the goal was not reached the refund function from the vault contract is called refunding all the contributed eth.

claimUnsold

If the sale has ended, and there are still tokens belonging to the crowdsale contract they will be transferred to the function caller using a function called transferTokens.

Suggestions

1. Use default ERC20 transfer function instead of custom transferTokens function.

updateWhitelist

Sets the status for a passed list of addresses in the whitelist mapping.

finalize

Once the sale has ended, this function can be called once. If the sales goal was reached it unpauses the token contract, transfers its ownership and closes the vault. If the goal was not reached, it will enable refunds. Finally it sets isFinalized to true.

Suggestions

1. Set isFinalized to true before any external calls.

forwardFunds

Sends funds to the vault contract using the deposit function.

hasEnded

Returns true if the sale has run out of time or cap was reached.

capReached

Returns true if weiRaised is greater than or equal to cap.

goalReached

Returns true if weiRaised is greater than or equal to goal.

isWhitelisted

Returns the value of whitelisted for a given address.

validPurchase

Ensures that current time is within sale range, ensures that Ether was sent with the transaction, and ensures that weiRaised is smaller than cap.

getBlockTimestamp

Returns the blocks current timestamp, function can be removed.

DataWalletToken.sol

Constructor

Sets the totalSupply of the token and sets the senders balance to the totalSupply.

transferTokens

This function allows the owner to transfer tokens. It was created due to the fact that the token is paused during the crowdsale.

Suggestions

1. Instead of creating a custom function the token contract should override the transfer function allowing the user to still transfer if he is the owner or crowdsale.