## Security Features

### Contract Design

Contracts need to be activated by the owner before most functions are allowed. The owner can place the contract into admin only mode to prevent further interactions if problems are suspected.

Exchange

Wallet

### Coin Balances and Ether

Coin balances are held in Token Data, the access to this is restricted to 2 specific contracts.

The ether owed to the ozcoin account is held in the Standard Token contract until withdrawn to the ozcoin account, I recommend this is done regularly to prevent large values building up in the contract. The ether owing to affiliates and affiliate companies is held in Standard Token until they chose to withdraw it.

The withdrawal of funds by affiliates and affiliate companies uses a recommended withdrawal pattern (<https://solidity.readthedocs.io/en/develop/common-patterns.html>) , to avoid callstack depth and reentrancy attacks.

Arbitration functionality allows invalid transactions to be reversed. To reverse a transaction requires a request from the original account, the ozcoin account holder must agree to this request then the arbiter can make the transaction. The actions by the arbiter and ozcoin account holder must occur within a certain time from the time of request.

### Account Changes

Changes to primary contract addresses and accounts are only possible when the contract is in admin only mode.

There is an administrator role for the user accounts, this can only be set up by the owner of the User contract.

All changes to accounts having special privileges produce events which can be monitored.

## Potential Attack Scenarios

### Attempts to steal ether from the contracts

Only affiliate and affiliate company balances are kept for any length of time, thus the contract is expected to only have a small ether balance, making it less of a target for attackers. Withdrawal of ether uses the recommended safe pattern.

### Attempts to steal coins

We are able to monitor all transactions and can reverse transactions if required.

There is the ability to make transactions go into a pending state if required, giving time to check any suspected transactions and revert them if needed.

Accounts can be frozen if necessary to prevent movement of coins.

### Denial of Service Attacks

The pattern used for withdrawing ether avoids the potential stack depth problem.

Although a malicious actor could generate spurious transfers between accounts that he holds this would achieve little, would cost the actor ether, and ozcoins in transfer fees.

### Arbitration procedure

If a user needs a transaction to be changed he can request arbitration via the Standard token requestArbitration function.

The arbiter then has to approve arbitration (TokenData approveArbitration function) within 500 blocks.

The ozcoin account then can move coins between accounts as necessary (TokenData aribtrateTransfer function) .

### Mitigating potential problems

If we suspect that a contract is compromised, I recommend the following steps :

* Stop further interactions from the web until the problem is resolved
* Put the contract into Admin mode, this will prevent much of the user interactions
* If a particular account is thought to be compromised or acting maliciously we can freeze the account from the Standard Token contract.

If in the worst case we are unable to resolve the situation with the current contracts :

* Record all users ether balances in Standard Token.
* Withdraw all ether from Standard Token into the ozcoin account - use the withdrawAllEther function (you need to make Standard Token active for this function)
* Create a new Token Data contract specifying the old Token Data as the parent, the coin balances for each account can then be copied across. The contracts will then need to be rewired to the new TokenData contract.
* Repay the user balances from the ozcoin account