High severity issues

- **One user *owner has full control / Centralization**

In the functions below, the owner role has access to make any change!

- ✓ function excludeAccountRewards(address account) external onlyOwner
- ✓ function excludeAccountFee(address account) external onlyOwner
- ✓ function setLockContract(address lockContract) external onlyOwner
- ✓ function setSecondLockContract(address lockContract) external onlyOwner
- ✓ function withdrawStuckToken(address recipient, address token)
- ✓ function setFeeHolders(uint8 feeamt) external onlyOwner
- ✓ function includeAccountRewards(address account) external onlyOwner
- ✓ function includeAccountFee(address account) external onlyOwner

```
1     // ****** Owner can exclude any wallet and account from get reward !
2     function excludeAccountRewards(address account) external onlyOwner {
3         require(
4             !_isExcluded[account],
5             "Account is already excluded from rewards"
6         );
7         if (_rOwned[account] > 0) {
8             _tOwned[account] = tokenFromReflection(_rOwned[account]);
9         }
10        _isExcluded[account] = true;
11        _excluded.push(account);
12    }
13
```

- ✓ **Recommendation**

The risk describes the current project design and potentially makes iterations to improve the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices; multi-signature wallets.

Low severity issues

- **Unlocked Pragma Used**

Contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

```
// bad
pragma solidity ^0.4.4;


// good
pragma solidity 0.4.4;
```

✓ Recommendation

Should lock pragmas to a specific compiler version.

- **Missing Emit Events**

There should always be events emitted in the sensitive functions that are controlled by centralization roles. There is not any event emitting in the below functions.

- function excludeAccountRewards(address account) external onlyOwner
- function excludeAccountFee(address account) external onlyOwner
- function setLockContract(address lockContract) external onlyOwner
- function setSecondLockContract(address lockContract) external onlyOwner
- function withdrawStuckToken(address recipient, address token)
- function setFeeHolders(uint8 feeamt) external onlyOwner
- function includeAccountRewards(address account) external onlyOwner
- function includeAccountFee(address account) external onlyOwner

✓ Recommendation

S It is recommended emitting events for the sensitive functions that are controlled by the centralization role should lock pragmas to a specific compiler version.