https://etherscan.io/token/0x5891599664Ed15C6e88041B4F5bc08594f026f0e#code

Medium severity issues

• Initial Token Distribution / Centralization

In the constructor function, the deployer is minting tokens for his wallet address. All the initial supply are sent to the contract deployer when deploying the contract. This could be a centralization risk, the deployer can distribute those tokens without obtaining the consensus of the community.

```
451     constructor(string memory name, string memory symbol, uint8
452       _name = name;
453       _symbol = symbol;
454       _decimals = decimals;
455
456       // set tokenOwnerAddress as owner of all tokens
457       _mint(tokenOwnerAddress, totalSupply);
458
459       // pay the service fee for contract deployment
460       feeReceiver.transfer(msg.value);
461     }
462
```

✓ Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key, We also advise the client to adopt Multisig, Timelock, and/or DAO in the project to manage the specific account in this case.

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.