# Zilliqa Token Audit

Zero Knowledge Labs Auditing Services

**AUTHOR: MATTHEW DI FERRANTE** 

# **Audited Material Summary**

The audit consists of the following contracts:

```
ded84c783b29ff583a39d49750da2bf96696788ff06c89d907c4477713007290
ZilliqaToken.sol
```

The contract implements a ERC20 Token, with Pausable functionality. ERC20 and Pausable implementation are taken from Zeppelin's solidity code. It also allows token holders to burn their tokens, reducing the supply.

#### Security

There are no security issues in the code.

### ZilliqaToken.sol

The ZilliqaToken contract inherits from Zeppelin's PausableToken:

```
contract ZilliqaToken is PausableToken
```

The contract has only one custom modifier, validDestination, which ensures that the argument address is not zero nor the contract's address.

#### Constructor

```
function ZilliqaToken( address _admin, uint _totalTokenAmount )
2
           // assign the admin account
4
           admin = _admin;
5
           // assign the total tokens to zilliqa
6
7
           totalSupply = _totalTokenAmount;
8
           balances[msg.sender] = _totalTokenAmount;
9
           Transfer(address(0x0), msg.sender, _totalTokenAmount);
10
       }
```

The constructor sets the contract's admin, total Supply, and emits a Transfer address notifying a token creation event from  $0 \times 0$  to msg. sender.

#### transfer

```
function transfer(address _to, uint _value) validDestination(_to)
    returns (bool)

return super.transfer(_to, _value);
}
```

The transfer function overrides the standard ERC20 transfer to apply the validDestination modifier. All else remains the same.

#### transferFrom

The transferFrom function overrides the standard ERC20 transferFrom to apply the validDestination modifier. Like for transfer, all else remains the same.

#### burn

```
function burn(uint _value) returns (bool)
{
    balances[msg.sender] = balances[msg.sender].sub(_value);
    totalSupply = totalSupply.sub(_value);
    Burn(msg.sender, _value);
    Transfer(msg.sender, address(0x0), _value);
    return true;
}
```

The burn function allows a token holder to destroy their coins, reducing the total supply. SafeMath is used when manipulating balances so there are no arithmetic security issues.

On succes, a Burn event is emitted, as well as a Transfer event notifying a token transfer of \_value from msg.sender to 0x0.

#### burnFrom

```
function burnFrom(address _from, uint256 _value) returns (bool)
{
    assert( transferFrom( _from, msg.sender, _value ) );
    return burn(_value);
}
```

The burnFrom function is a helper that allows smart contracts calling the token contract to burn tokens in one call.

# emergencyERC20Drain

```
function emergencyERC20Drain( ERC20 token, uint amount ) onlyOwner {
    // owner can drain tokens that are sent here by mistake
    token.transfer( owner, amount );
}
```

This function allows the contract owner to claim and rescue arbitrary ERC20 tokens sent to this contract by mistake.

# **Disclaimer**

This audit concerns only the correctness of the Smart Contracts listed, and is not to be taken as an endorsement of the platform, team, or company.

# **Audit Attestation**

This audit has been signed by the key provided on https://keybase.io/mattdf - and the signature is available on https://github.com/mattdf/audits/

Author: Matthew Di Ferrante 5