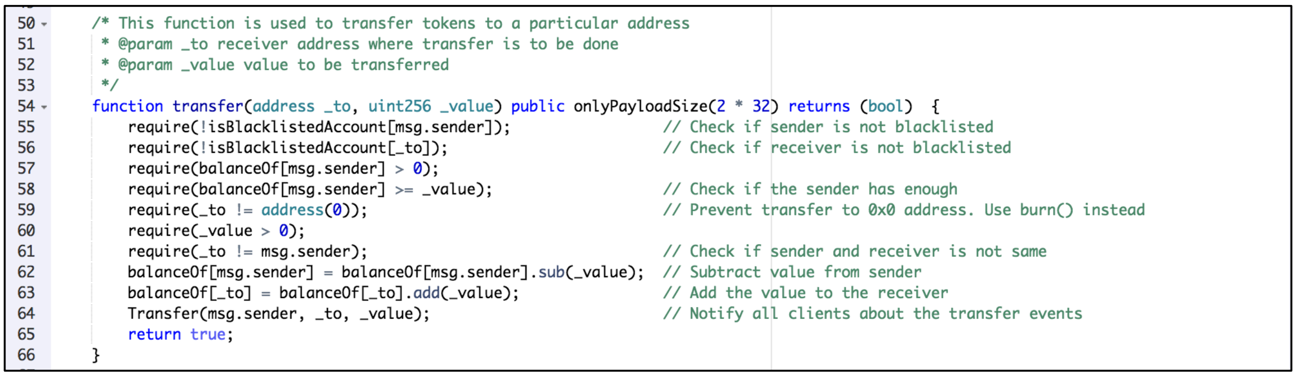
ICO Experts Agency

Code audit for MorphToken

1. Method `transfer`

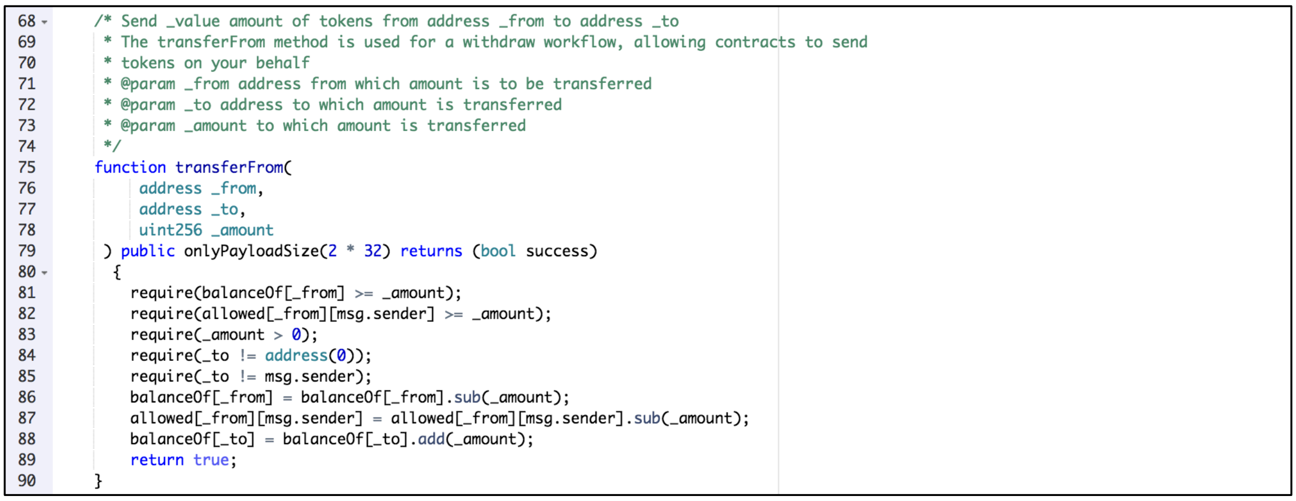


```
50    /* This function is used to transfer tokens to a particular address
51     * @param _to receiver address where transfer is to be done
52     * @param _value value to be transferred
53     */
54    function transfer(address _to, uint256 _value) public onlyPayloadSize(2 * 32) returns (bool)  {
55        require(!isBlacklistedAccount[msg.sender]);                // Check if sender is not blacklisted
56        require(!isBlacklistedAccount[_to]);                       // Check if receiver is not blacklisted
57        require(balanceOf[msg.sender] > 0);
58        require(balanceOf[msg.sender] >= _value);                  // Check if the sender has enough
59        require(_to != address(0));                                // Prevent transfer to 0x0 address. Use burn() instead
60        require(_value > 0);
61        require(_to != msg.sender);                                // Check if sender and receiver is not same
62        balanceOf[msg.sender] = balanceOf[msg.sender].sub(_value);  // Subtract value from sender
63        balanceOf[_to] = balanceOf[_to].add(_value);               // Add the value to the receiver
64        Transfer(msg.sender, _to, _value);                         // Notify all clients about the transfer events
65        return true;
66    }
```

minor
code within **rows 57-58** can be deleted without loss of security and validity, because the same will be checked within code in the **row 62**

minor
code within **row 60** can be deleted without loss of security and validity, because the same will be checked within code in the **row 63**

2. Method `transferFrom`



```
68    /* Send _value amount of tokens from address _from to address _to
69     * The transferFrom method is used for a withdraw workflow, allowing contracts to send
70     * tokens on your behalf
71     * @param _from address from which amount is to be transferred
72     * @param _to address to which amount is transferred
73     * @param _amount to which amount is transferred
74     */
75    function transferFrom(
76          address _from,
77          address _to,
78          uint256 _amount
79     ) public onlyPayloadSize(2 * 32) returns (bool success)
80      {
81        require(balanceOf[_from] >= _amount);
82        require(allowed[_from][msg.sender] >= _amount);
83        require(_amount > 0);
84        require(_to != address(0));
85        require(_to != msg.sender);
86        balanceOf[_from] = balanceOf[_from].sub(_amount);
87        allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_amount);
88        balanceOf[_to] = balanceOf[_to].add(_amount);
89        return true;
90    }
```

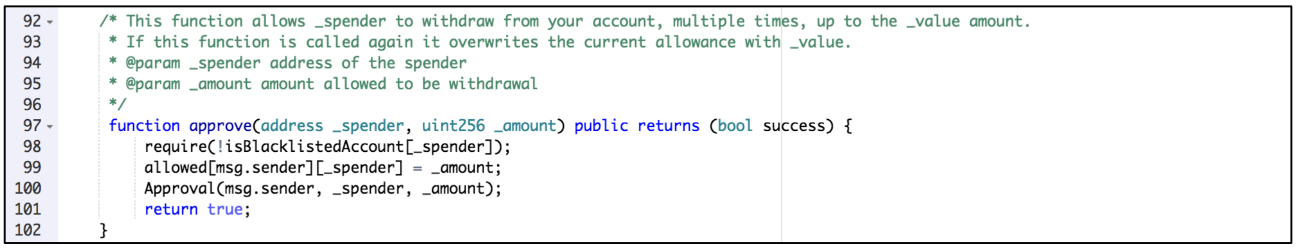
minor
code within **row 81** can be deleted without loss of security and validity, because the same will be checked within code in the **row 86**

minor
code within **row 83** can be deleted without loss of security and validity, because the same will be checked within code in the **row 88**

critical

in the **row 79** it should be **onlyPayloadSize(3 * 32)** instead of **onlyPayloadSize(2 * 32)** otherwise this function will crash on invoking it

3. Method `approve`



```
 92     /* This function allows _spender to withdraw from your account, multiple times, up to the _value amount.
 93      * If this function is called again it overwrites the current allowance with _value.
 94      * @param _spender address of the spender
 95      * @param _amount amount allowed to be withdrawal
 96      */
 97      function approve(address _spender, uint256 _amount) public returns (bool success) {
 98          require(!isBlacklistedAccount[_spender]);
 99          allowed[msg.sender][_spender] = _amount;
100          Approval(msg.sender, _spender, _amount);
101          return true;
102     }
```

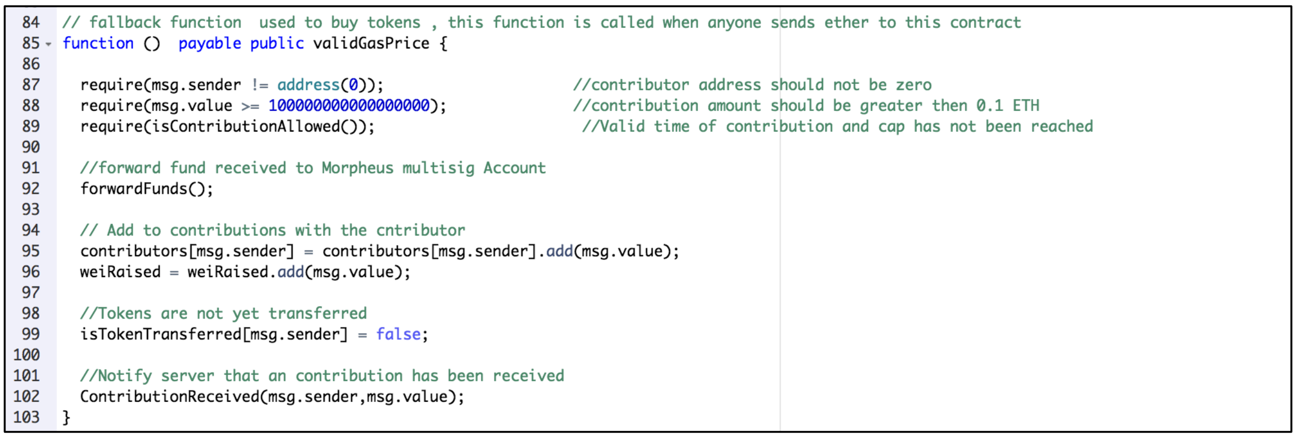
critical

there is verification in the **row 81** that spender is not in a black list, while there is no verification that token owner who is giving allowance to transfer his tokens, is not in a **Black List**

use case

- **owner** block **address1** by adding it to a **Black List**
- **address1** use method approve to let address2 allowance to transfer his tokens
- **address2** use method **transferFrom** to transfer tokens used by **address1**

4 fallback function

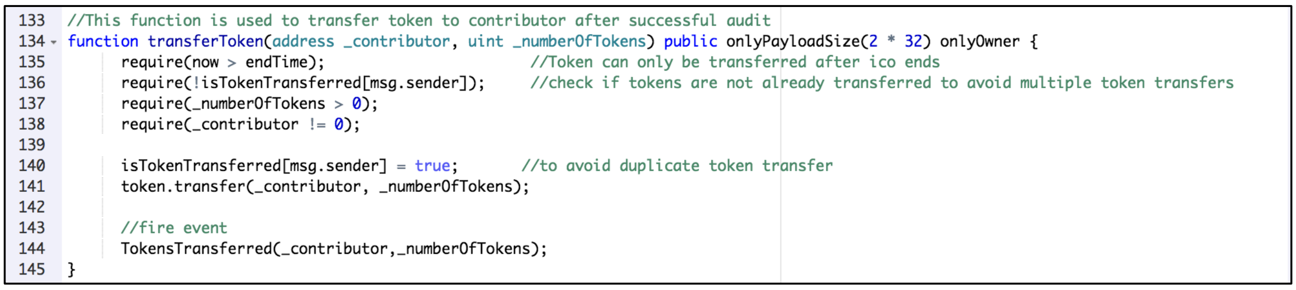


```
 84 // fallback function  used to buy tokens , this function is called when anyone sends ether to this contract
 85 function ()  payable public validGasPrice {
 86 
 87   require(msg.sender != address(0));                    //contributor address should not be zero
 88   require(msg.value >= 100000000000000000);             //contribution amount should be greater then 0.1 ETH
 89   require(isContributionAllowed());                      //Valid time of contribution and cap has not been reached
 90 
 91   //forward fund received to Morpheus multisig Account
 92   forwardFunds();
 93 
 94   // Add to contributions with the cntributor
 95   contributors[msg.sender] = contributors[msg.sender].add(msg.value);
 96   weiRaised = weiRaised.add(msg.value);
 97 
 98   //Tokens are not yet transferred
 99   isTokenTransferred[msg.sender] = false;
100 
101   //Notify server that an contribution has been received
102   ContributionReceived(msg.sender,msg.value);
103 }
```

minor

for the **row 88** instead of using **100000000000000000** it will be better to use **0.1 ether**

5. Method `transferToken`



```
//This function is used to transfer token to contributor after successful audit
function transferToken(address _contributor, uint _numberOfTokens) public onlyPayloadSize(2 * 32) onlyOwner {
        require(now > endTime);                           //Token can only be transferred after ico ends
        require(!isTokenTransferred[msg.sender]);       //check if tokens are not already transferred to avoid multiple token transfers
        require(_numberOfTokens > 0);
        require(_contributor != 0);

        isTokenTransferred[msg.sender] = true;          //to avoid duplicate token transfer
        token.transfer(_contributor, _numberOfTokens);

        //fire event
        TokensTransferred(_contributor,_numberOfTokens);
}
```

major
this method has no invocation nowhere in the code

major
even in case this function will be used somewhere, its logic is incorrect: if it will be used and not all tokens of current user will be transferred to him, further transferring for him will be locked

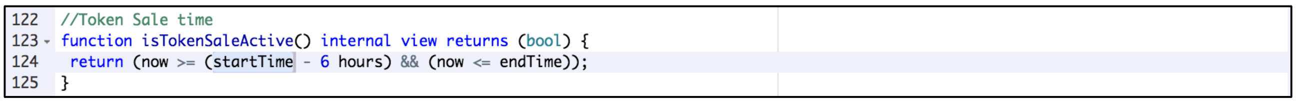
6. Method `setMaxGasPrice`, variable `maxGasPrice` and modifier `validGasPrice`

Usage of all these is arising lots of questions concerning necessity of doing this

7. Detected functionality according to information from landing page https://morpheus.network/token/

As far as there were no concrete technical requirements for audited smart contracts besides the following information from landing page https://morpheus.network/token/:

presale start date: March 2nd, 2018 at 12:00 pm EST – present in smart contract
presale duration: 10 days – present in smart contract
presale token price: $ 0.5 = 1.12 MORPH (12% bonus) – absent in smart contract
minimum presale contribution: 0.1 ETH – present in smart contract
soft cap min distribution: 21.95 million morph tokens ($ 1,800,000) – absent in smart contract
hard cap max distribution: 100 million morph tokens ($ 36,000,000) – absent in smart contract
main sale duration: 30 days – present in smart contract
main sale date: to be announced – absent in smart contract



```
122 //Token Sale time
123 function isTokenSaleActive() internal view returns (bool) {
124   return (now >= (startTime - 6 hours) && (now <= endTime));
125 }
```

according to **row 124** token sale should be active 6 hours before presale start date (March 2nd, 2018 at 12:00 pm EST) which is not mentioned on landing page (which was considered as requirements for smart contract)

Conclusion

Current state of audited smart contracts is recognized to be so that they **can not be used for ICO purposes** until the following issues are not eliminated:

1. smart contracts do not include about half of required logic (technical requirements)

2. smart contracts do not include ICO functionality (ability for users to buy tokens etc.)

3. smart contracts have major and critical issues